Chromatics™
NEW DIMENSIONS IN COLOR GRAPHICS

PRELIMINARY

# OPERATOR'S MANUAL

## CG SERIES
## COLOR GRAPHICS COMPUTERS

**MODELS**

| | | |
|---|---|---|
| 1398 | 1598 | 1998 |
| 1399 | 1599 | 1999 |

100168
REVISED 11/78

**Chromatics, Inc.** / 3923 Oakcliff Industrial Court / Atlanta, Georgia 30340 / Telephone 404/447-8797

# WARNING

Before connecting any external cables to your CG terminal, check the I/O connector pin assignments found in Appendix D of your Operator's Manual. Note that the undefined pins of the RS-232 connectors (pins 10, 11 & 25) have been used for the TTL I/O lines. Connection to any external signals on these pins which are not compatible will result in PERMANENT damage to your unit and will not be covered by the standard warranty. When connecting an "unknown" RS-232 cable to these ports, it is best that any [redacted] be cut which are connected to these three pins.

TABLE OF CONTENTS

TABLE OF CONTENTS - continued

TABLE OF CONTENTS - continued

# 1. INTRODUCTION

This operator's manual gives detailed information on the use of the Chromatics CG series of intelligent color display terminals. All standard and optional features are covered, but the details of certain optional processors, (DOS, Text Editor, Z-80 Assembler, the PROM Programmer and BASIC), are omitted, since these appear in separate manuals.

This chapter covers general information about the CG terminals and introduces terminology used in the rest of the manual. Chapter 2 explains the system orginization - the basic configuration. The third chapter is a thorough guide to the standard system software, while remaining chapters are devoted to the various optional features.

## 1.1 General System Description

The Chromatics CG series of terminals are complete, self-contained, ultra-high resolution color graphics terminals with integral Z-80 microprocessor, memory and input/output (I/O). Each dot on the 512 by 256 (CG 1398, 1598, 1998) or 512 by 512 (CG 1399, 1599, 1999) screen may be individually set to one of eight colors and may be set to blink at 1.9 Hz. The extremely flexible high resolution display capabilities of these terminals makes them suitable for alphanumeric as well as graphics applications. The built-in microprocessor provides computing power for stand-alone operation as well as intelligent terminal use.

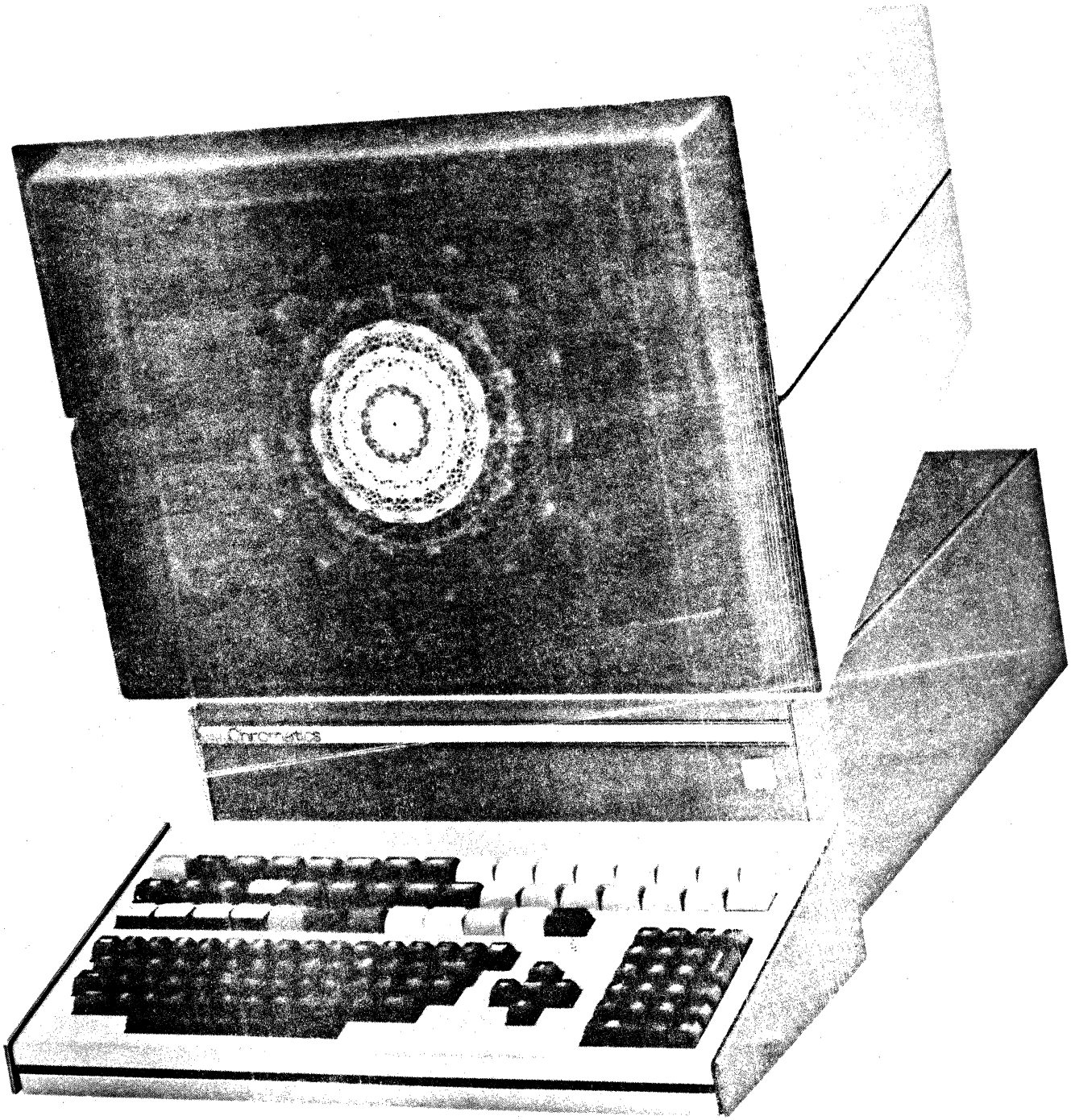


FIGURE 1.1

Each terminal is contained in a single package, attractively styled for stand-alone operation or integration into existing equipment, (see Figure 1.1). There are no external controllers, interface cards or power supplies to be mounted or hidden in equipment racks. All components are easily accessible to allow fast and efficient servicing.

## 1.2 Initial Checkout and Start-up

Before beginning operation, the terminal should be situated on a level surface of convenient height, such as a table or desk. Note that the detachable keyboard allows flexible placement for ease of use. The terminal should, of course, be placed so that it is improbable that it will be subject to external damage or spills. The keyboard should be attached to the terminal by connecting its flat cable to the socket marked J8 on the back of the terminal. The terminal may now be plugged into any standard three-prong wall outlet, (105-125 volts, 60 Hz, 600 watts). (The terminal may be configured for 205-250 volt operation - Option 11, and/or 50 Hz - Option 12).

### 1.2.1 The POWER Key

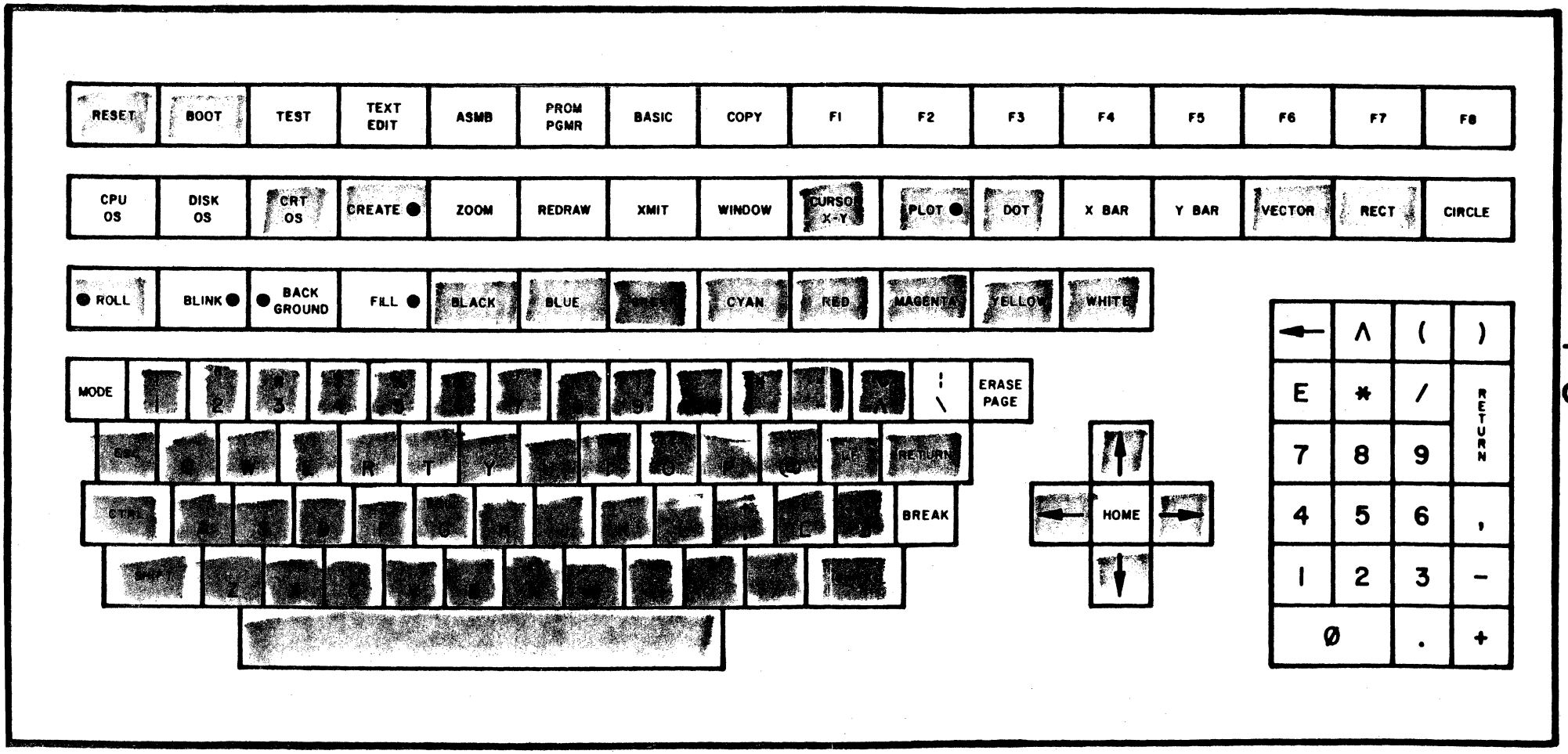
The illuminated POWER key is located below and to the right of the screen on the screen cabinet, (see Figure 1.1). The POWER key is used to turn the terminal on and off, and remains lit at all times when the terminal is on. When the terminal is turned on, all components are brought to full power and a function equivalent to the BOOT key, (see section 2.4.1), is executed.

## 1.3 Introduction to the Keyboard

The keyboard is the primary input device for the terminal. Each key, (except the RESET key - see section 2.4.1), generates a coded eight bit byte when struck. If the key is held for about 3/4-ths of a second, the byte is repeated at the rate of 20 Hz. The eight bit codes from the keyboard are interpretted by a special keyboard handler routine and converted into a sequence of one or more seven bit ASCII codes to be used by the operating program. (The BREAK key is an exception to this rule - see section 3.3.5). The meaning of these ASCII codes is the subject of much of the rest of this manual.

### 1.3.1 Key Groups

There are four groups of keys on the 128-key keyboard. The upper three rows are the special function keys. These are used for functions which are especially defined for the Chromatics CG series. Certain keys will not be operational unless the corresponding options have been purchased. The numeric keypad is the block of keys on the lower right. This block is arranged in standard adding machine format to allow easy entry of numeric information and equations. The group of five keys to the left of the numeric keypad is the cursor control. The cursor control is used to allow easy positioning of the cursor anywhere on the screen. The final group of keys is the basic keyboard, which is the typewriter-like block of keys on the lower left. With the exception of the RESET key, all other functions on the keyboard can be produced on the basic keyboard by a combination of one or more keys. The additional keys allow fast and accurate keying of frequently used functions.




NOTE: ● INDICATES ILLUMINATED KEY.

FIGURE 1.2

### 1.3.2 ASCII

The seven bit codes ultimately generated from keyboard input are standard ANSI ASCII codes, but some of the control codes have been defined for special purposes for the CG terminal series. The correspondence between the basic keyboard keys and the codes generated is shown in Figure 1.3. Notice that the CTRL and SHIFT keys are used in combination with other keys to generate some of the codes. CTRL and SHIFT do not generate codes by themselves, but rather serve as modifiers for other keys. The CTRL and SHIFT keys must be held down while the key to be modified is struck. This action will be indicated in this manual by the name of the modifier(s) followed by the name of the key modified. For example, SHIFT A will generate the code for a lower case A. (This usage is convenient for most terminal applications, although it is the reverse of typewriter convention. If desired, the meaning of SHIFT can be switched for alphabetic characters).

The white keys in Figure 1.4 will be called primary keys. These 47 keys can, in combination with the CTRL and SHIFT keys, generate any of the 128 ASCII codes. The meaning of shifting the non-alphabetic keys is indicated on the top half of each key. For example, SHIFT 4 is equivalent to $. The symbols on the upper half of the alphabetic keys indicate the corresponding control function. For example, CTRL G is equivalent to BELL. (BELL causes an audible tone to be generated when sent). Most control functions which are labelled on the primary keys can be keyed using special function keys. The following functions can only be keyed using the CTRL key:

# ASCII CODE ASSIGNMENT

| HEX | A3 | A2 | A1 | A0 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | A7 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | | | A6 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| | | | | A5 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| | | | | A4 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |
| 0 | 0 | 0 | 0 | 0 | CONTROL @ TO 0 | CONTROL P TO _ | SHIFT @ TO ! | SHIFT , TO \ | | | SHIFT @ TO 0 | SHIFT P TO _ | CONTROL @ TO 0 | CONTROL P TO _ | | | | | | |
| 1 | 0 | 0 | 0 | 1 | NULL 0 | | SPACE 32 | 0 48 | @ 64 | P 80 | ` 96 | p 112 | | | SPACE 32 | 48 | 64 | 80 | 96 | 112 |
| 2 | 0 | 0 | 1 | 0 | MODE 1 | | ! 33 | 1 49 | A 65 | Q 81 | a 97 | q 113 | | | 33 | 49 | 65 | 81 | 97 | 113 |
| 3 | 0 | 0 | 1 | 1 | | | " 34 | 2 50 | B 66 | R 82 | b 98 | r 114 | | | ↓ 34 | 50 | 66 | 82 | 98 | 114 |
| 4 | 0 | 1 | 0 | 0 | | | # 35 | 3 51 | C 67 | S 83 | c 99 | s 115 | | | ↑ 35 | 51 | 67 | 83 | 99 | 115 |
| 5 | 0 | 1 | 0 | 1 | ONE DOT UP 5 | MODE CANCEL 21 | $ 36 | 4 52 | D 68 | T 84 | d 100 | t 116 | | | ← 36 | 52 | 68 | 84 | 100 | 116 |
| 6 | 0 | 1 | 1 | 0 | DELETE CHARACTER 6 | ONE DOT DOWN 22 | % 37 | 5 53 | E 69 | U 85 | e 101 | u 117 | | | → 37 | 53 | 69 | 85 | 101 | 117 |
| 7 | 0 | 1 | 1 | 1 | BELL 7 | INSERT CHARACTER 23 | & 38 | 6 54 | F 70 | V 86 | f 102 | v 118 | | | ↓ 38 | 54 | 70 | 86 | 102 | 118 |
| 8 | 1 | 0 | 0 | 0 | BS 8 | | ' 39 | 7 55 | G 71 | W 87 | g 103 | w 119 | | | ↑ 39 | 55 | 71 | 87 | 103 | 119 |
| 9 | 1 | 0 | 0 | 1 | TAB 9 | ONE DOT LEFT 25 | ( 40 | 8 56 | H 72 | X 88 | h 104 | x 120 | | | ← 40 | 56 | 72 | 88 | 104 | 120 |
| A | 1 | 0 | 1 | 0 | LF 10 | | ) 41 | 9 57 | I 73 | Y 89 | i 105 | y 121 | | | → 41 | 57 | 73 | 89 | 105 | 121 |
| B | 1 | 0 | 1 | 1 | VT 11 | ESC 27 | * 42 | : 58 | J 74 | Z 90 | j 106 | z 122 | | | 42 | 58 | 74 | 90 | 106 | 122 |
| C | 1 | 1 | 0 | 0 | ERASE PAGE 12 | HOME 28 | + 43 | ; 59 | K 75 | [ 91 | k 107 | { 123 | | | 43 | 59 | 75 | 91 | 107 | 123 |
| D | 1 | 1 | 0 | 1 | CR 13 | CURSOR RIGHT 29 | , 44 | < 60 | L 76 | \ 92 | l 108 | \| 124 | | | 44 | 60 | 76 | 92 | 108 | 124 |
| E | 1 | 1 | 1 | 0 | A7 ON 14 | EOF 30 | - 45 | = 61 | M 77 | ] 93 | m 109 | } 125 | | | 45 | 61 | 77 | 93 | 109 | 125 |
| F | 1 | 1 | 1 | 1 | A7 OFF 15 | ONE DOT RIGHT 31 | . 46 | > 62 | N 78 | ^ 94 | n 110 | ~ 126 | | | 46 | 62 | 78 | 94 | 110 | 126 |
| | | | | | / 47 | ? 63 | 0 79 | _ 95 | o 111 | DEL 127 | | | | | 47 | 63 | 79 | 95 | 111 | 127 |

SAME AS 0 THRU 31

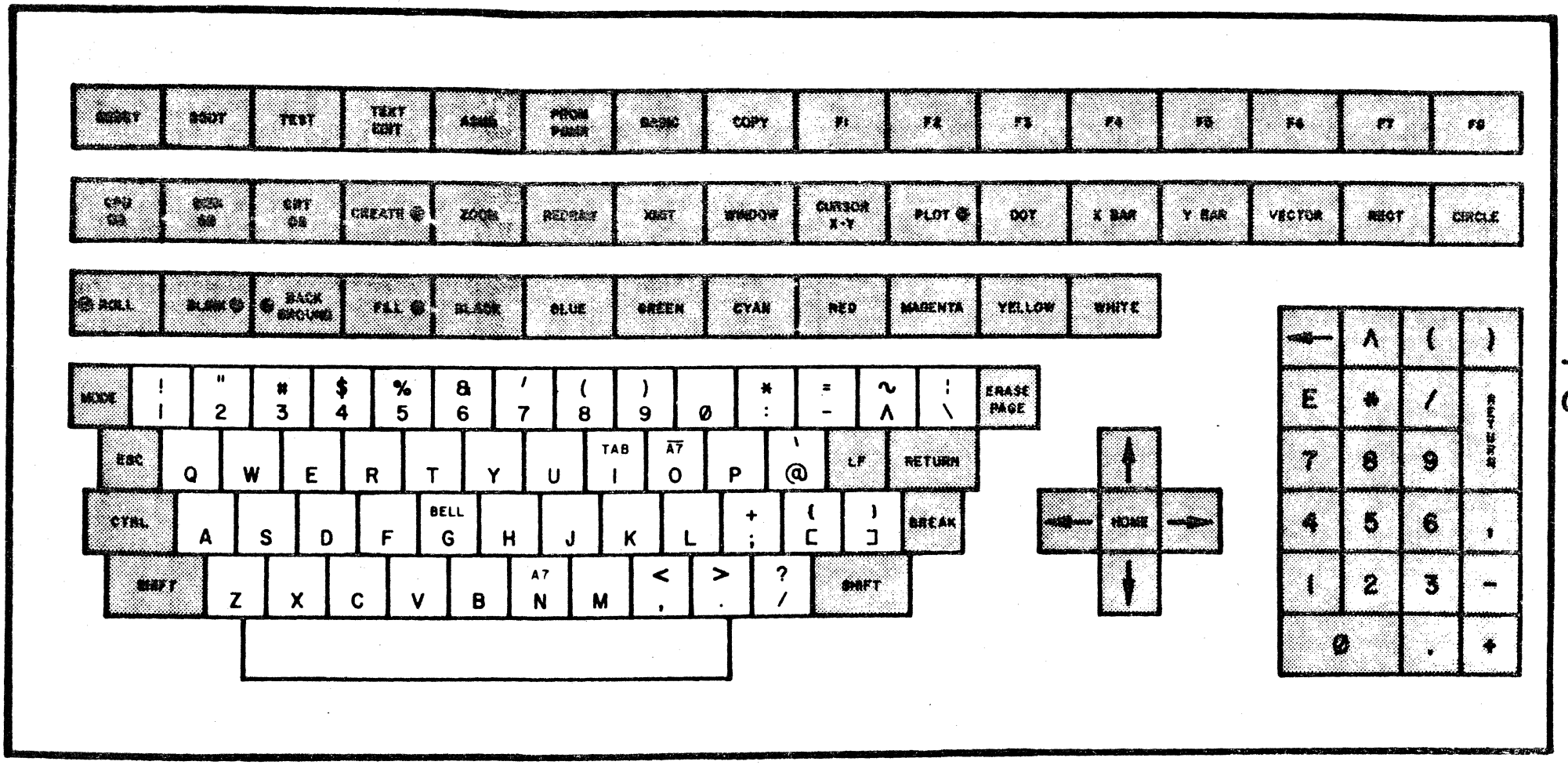
A7 ON IMPLIES SPECIAL GRAPHIC CHARACTERS.

NOTE: ENTRIES WITH ✕ INDICATE UNUSED ANSI ASCII CODES.

FIGURE 1.3

# KEYBOARD LAYOUT

## PRIMARY KEYS




NOTE: ● INDICATES ILLUMINATED KEY.

FIGURE 1.4

CTRL ^ = End of File (EOF)

CTRL W = Insert Character

CTRL F = Delete Character

There are also two codes which can only be keyed by using both the CTRL and SHIFT keys in combination with the key indicated:

SHIFT CTRL / = _ (underline)

SHIFT CTRL O = DEL (the delete symbol)

(Note: DEL is a non-printing symbol)

Finally, there are four functions which use the SHIFT key in combination with the cursor control keys:

SHIFT → = Move cursor one dot right

SHIFT ↓ = Move cursor one dot down

SHIFT ← = Move cursor one dot left

SHIFT ↑ = Move cursor one dot up

## 1.4 Terminology and Conventions

Throughout this manual, key sequences will be discussed. In order to indicate clearly and precisely the manner in which the keys are to be struck, the following conventions will be used.

1) Each key will be identified by the name used in Figure 1.2. Since some keys have alphabetic names, the individual alphabetic characters will be separated from adjacent key names with spaces. Blank characters to be generated by the space bar will always be explicitly indicated by "SPACE".

2) The key modifiers, CTRL and SHIFT, will immediately precede the single key which they are to modify.

3) Keys are to be struck in order from left to right. Several lines may be used to give a key sequence if necessary, but all returns and line feeds will be given explicitly by RETURN and LF, respectively.

4) All zeroes will be slashed (Ø) and alphabetic O's will be unslashed.

5) Underlined, lower case words will be used to name one of a set of keys or key sequences. This will be useful for brevity. For example, color can be defined to be any one of the color keys by

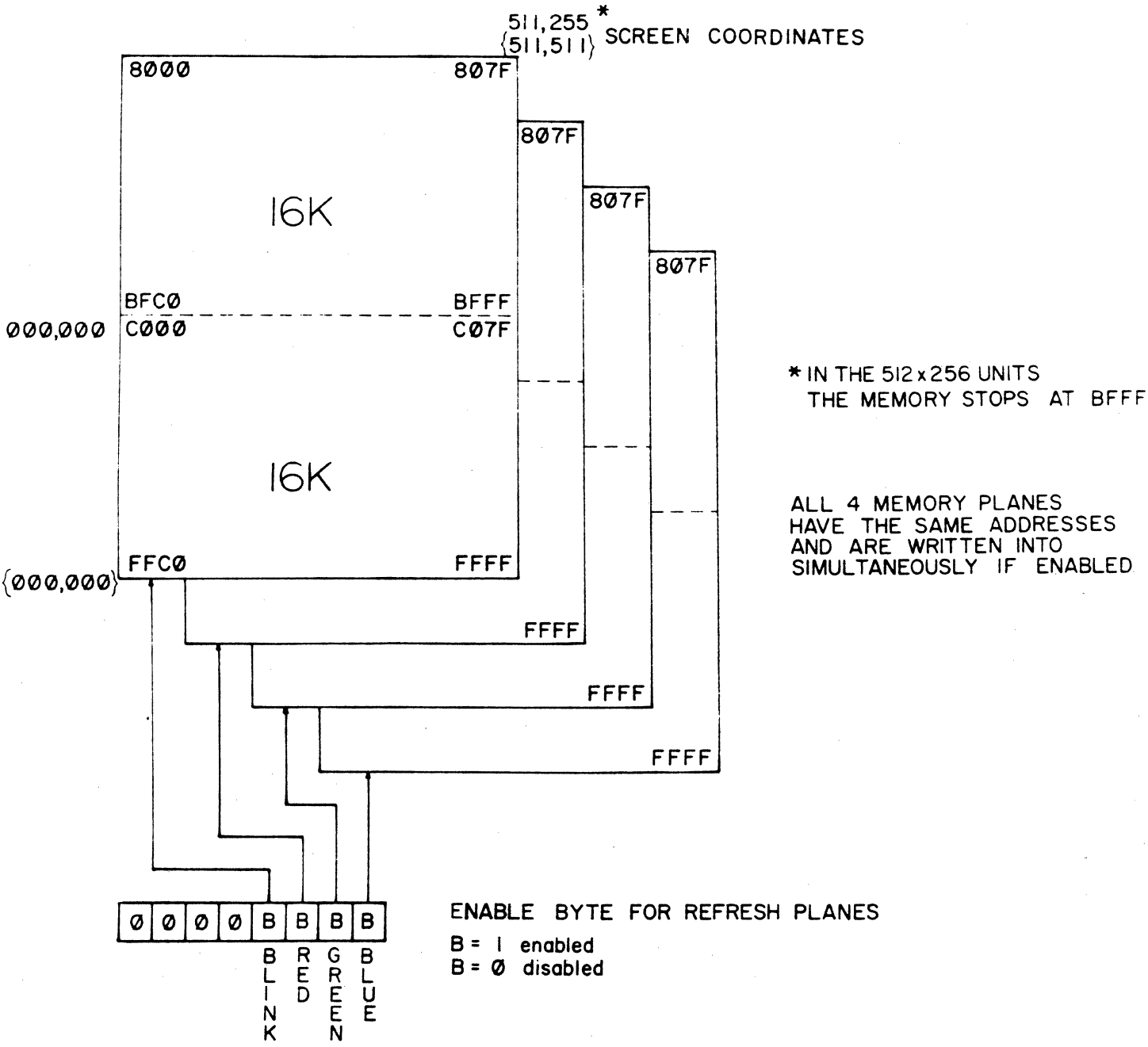
color ::= RED | GREEN | BLUE | YELLOW | CYAN | MAGENTA | WHITE | BLACK

The symbol "::=" should be read as "is defined by", and the vertical bars indicate a selection of one of the list.

Referring again to Figure 1.3, notice that the hexadecimal (base 16) and binary (base 2) representations of the Chromatics character set are indicated above and to the left of the main body of the table. These codings may be very important in communicating with another computer. For convenience, the hexadecimal coding of each function will be given at the point at which the function is discussed. Hexadecimal codes are given as two hex digits followed by an H. The first hex digit is given in the top row of Figure 1.3, and the second is given in the leftmost column. For example, the hexadecimal code for the letter Z is 5AH. When multiple characters are required for a function, the hexadecimal codes will be separated by commas, since there is no danger of confusion with the key named ",". Note that the hexadecimal codings given for the key sequences are not an alternate method of keying the information, but are an aid in generating key sequence codes from other computers.

# FIG. 2.05

ORGANIZATION OF THE REFRESH MEMORY



# 2. SYSTEM ORGANIZATION

The Chromatics CG Series of terminals are complete, stand-alone micro computers. The minimum system consists of a Z-80 microprocessor, a high-speed memory, and input/output (I/O) facilities using the keyboard and the CRT screen. This chapter will explain how this system is organized and the basics of its operation.

## 2.1 Physical Organization

Figure 2.1 illustrates the physical relationships between the major components of the system. The Z-80 microprocessor is the central component, controlling all the other components. All data flow passes through the Z-80. (Exception: the DMA - direct memory access - controller, Option 34, provides direct, high-speed data transfers at a rate up to 416,000 bytes per second).

There are two types of memory on the system: PROM and RAM. PROM, (programmable read only memory), is used to hold permanently loaded system software and tables. PROM cannot be overwritten and so provides protection against programming errors which might destroy the system software. All PROM provided by Chromatics is erasable by UV radiation; it can be erased and rewritten with the aid of the PROM Programmer - Option 52.

RAM is random access memory. It may be freely written by the software. RAM is used to store information which is variable during execution and may also be used to store programs. A large block of RAM is used as refresh memory for the CRT screen. Each dot on the screen corresponds to a location in the refresh memory which indicates its color.

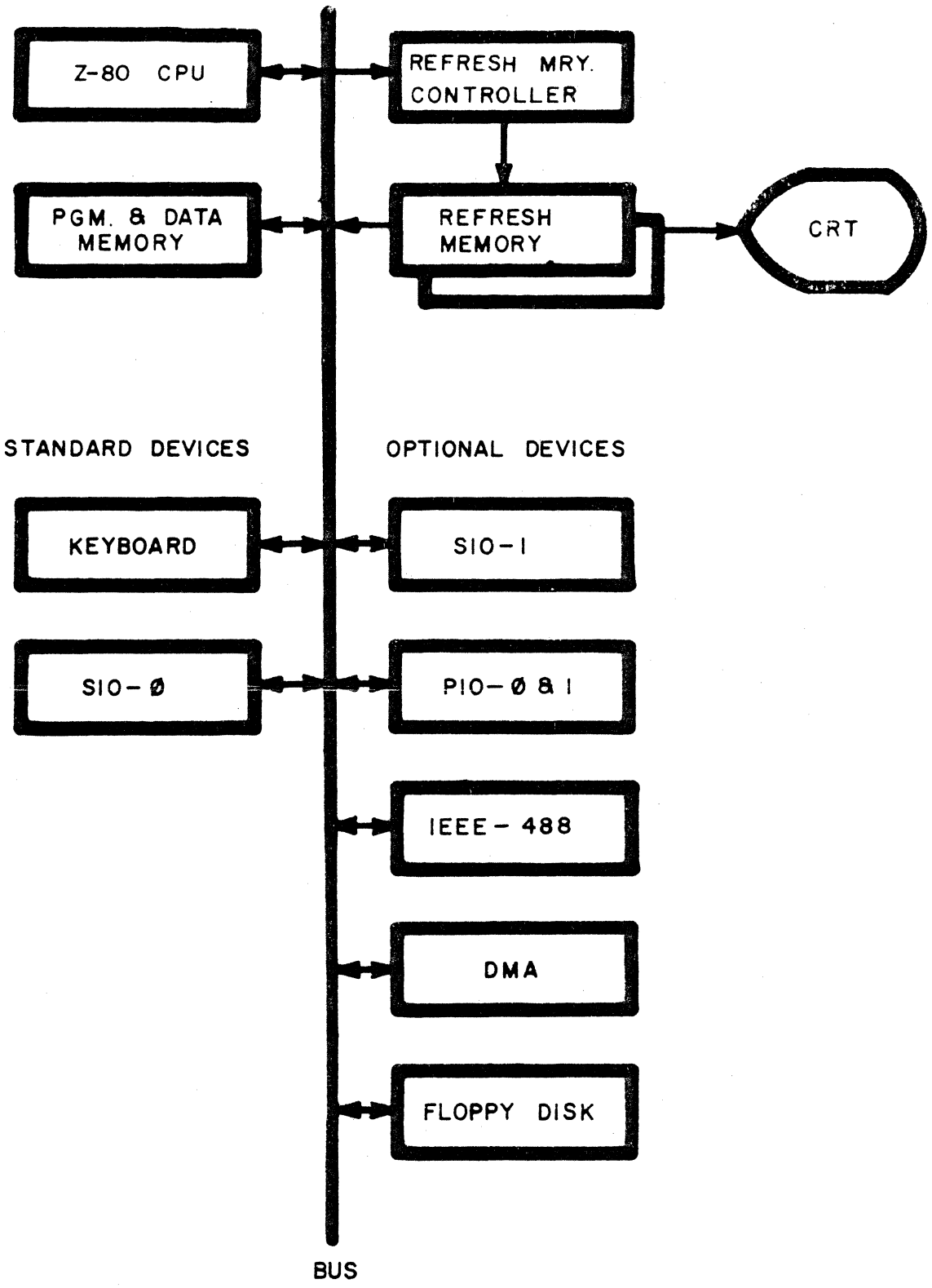



FIGURE 2.1 PHYSICAL ORGANIZATION

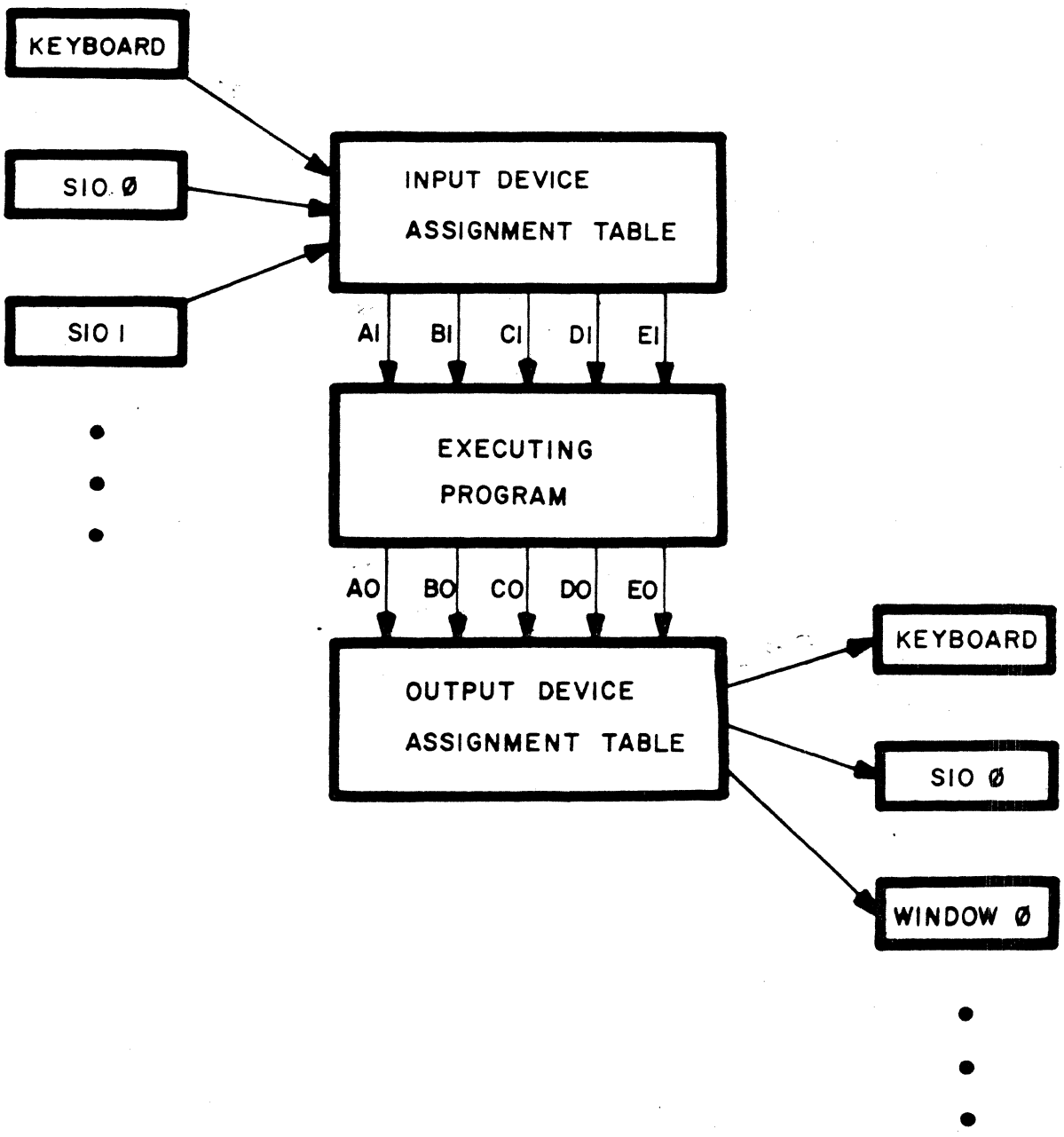



FIGURE 2.2 LOGICAL SYSTEM ORGANIZATION

## 2.2 Logical Organization

The logical organization of the system is represented in Figure 2.2 The device assignment tables connect the physical devices to the logical device names used by the main program. Each logical device name may have two physical devices attached to it. One of these is variable and can be easily modified using CRTOS, (see section 3.3.4). The other is fixed and is preset when the system is initialized. Fixed assignments can be changed using CPUOS, (see Chapter 5). The initial fixed assignment attaches the keyboard to logical input device AI and to logical output device AO; the remaining logical devices do not have a fixed assignment. (Note: the keyboard is an output device so that the illuminated keys can be controlled). The device assignment tables allow maximum flexibility in the use of the system.

## 2.3 Escape Code Processing

Escape codes are sequences of characters beginning with the ESC character. These codes are used for high-level control functions in the CG terminal series. Escape code processing may be done on any input line; system software generally processes all user generated input, but user programs may optionally omit escape code processing, (see the assembler manual for further details). The logic of escape code processing is shown in Figure 2.3. The following discussion assumes that escape code processing is being done.

Whenever as ESC character is sensed on the input line being processed, the escape mode is set to active. The ESC character and all succeeding characters on the input line are processed by the escape code handler

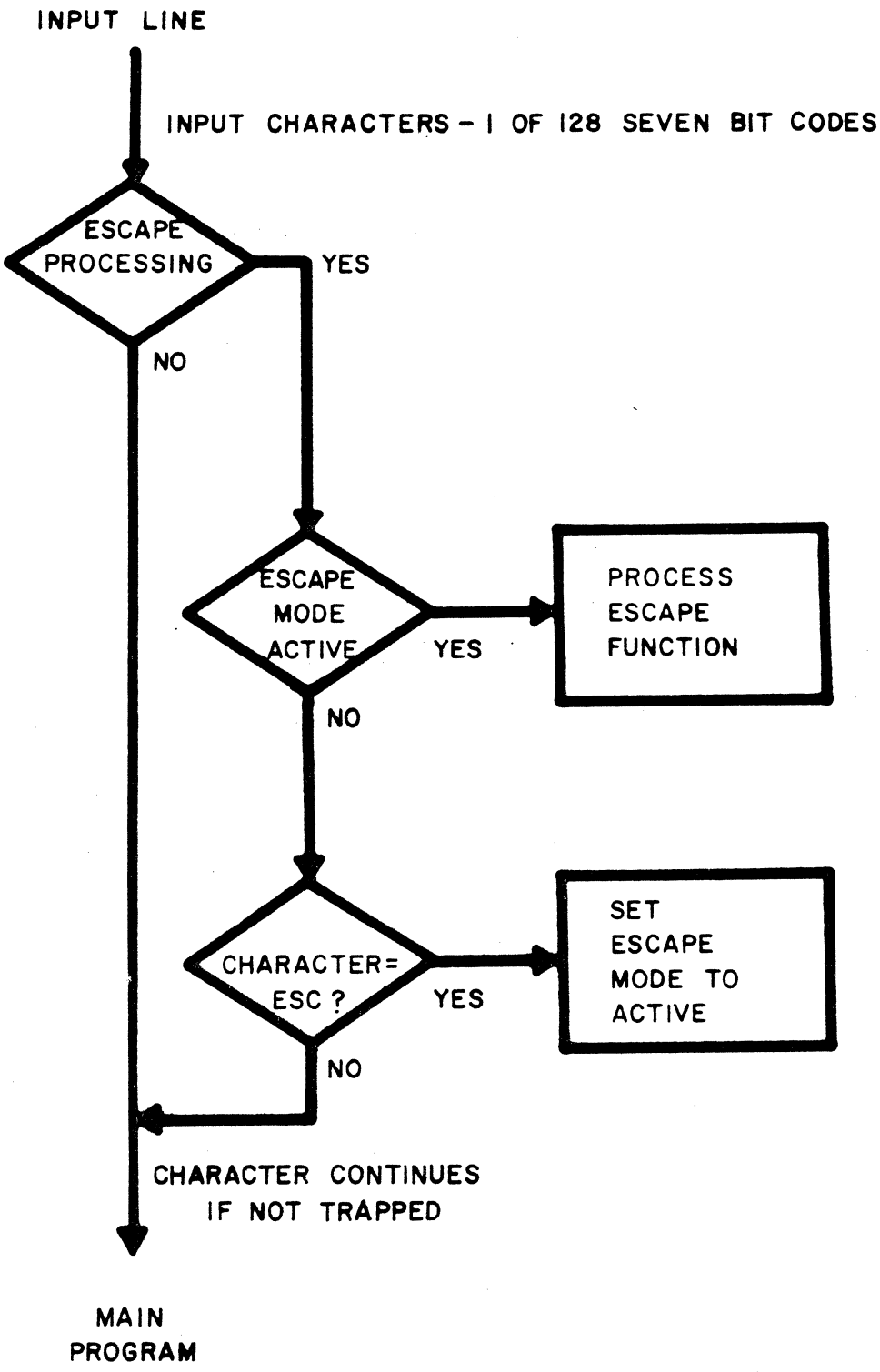



FIG. 2.3 ESCAPE CODE PROCESSING

as long as the escape mode is active. These characters are not processed by the main program. The number of characters diverted to the escape code handler depends on the escape function, which is determined by the first character following the ESC. The escape functions are discussed at appropriate places throughout the manual and are summarized in Appendix A. The last character of an escape code causes the escape mode to be reset to inactive so that normal processing can resume.

## 2.4 System Control Functions

The following escape code functions are used to transfer control between the various available main programs. This is done at the escape code level to allow easy "breaking out" of one program to go to another. If the corresponding optional features have not been purchased, these codes will result in no action. The operation of these functions is discussed more thoroughly in the sections devoted to the particular programs.

### 2.4.1 Cathode Ray Terminal Operating System (CRTOS)

The CRTOS is the basic operating program supplied with the standard system. There are three methods of entering CRTOS:

| | | |
|---|---|---|
| BOOT | ESC G | 1BH, 47H |

The BOOT Function causes all I/O devices and processor tables to be initialized to default conditions and the CRTOS main program to be executed. Depending on the state of the processor, a reset preceding the BOOT may be necessary for proper operation.

RESET

The RESET key is different from all other keys on the keyboard. It does not send an eight bit code to the keyboard handler, but instead

generates a hard-wired, master clear signal. Unlike BOOT, RESET does not reset I/O device and processor memory tables, but it does gain control, clear interrupts and execute CRTOS. RESET may be useful in recapturing control from a program in an infinite loop without losing information that may be helpful in debugging.

CRTOS | ESC T | 1BH, 54H

The CRTOS function causes the CRTOS main program to be executed with default device assignments, but does not initialize processor tables. For further details about these three commands, see Chapter 3.

### 2.4.2 Central Processing Unit Operating System (CPUOS) (Option 61)

CPUOS | ESC Z | 1BH, 5AH

The CPUOS function causes the CPU Operating System main program to begin execution with the current device assignments, (see Chapter 5).

### 2.4.3 BASIC Interpreter (Option 64)

BASIC | ESC B | 1BH, 42H

The BASIC function causes the BASIC Interpreter, (which is resident in PROM), to be initialized and to begin execution with current device assignments.

ESC E | 1BH, 45H

This function causes the BASIC Interpreter to be re-entered without initialization. See Chapter 6 for more detail.

### 2.4.4 Disk Operating System (DOS) (Option 41)

DISK OS | ESC D | 1BH, 44H

The DISK OS function causes the Disk Operating System (DOS) main program to begin execution with the current device assignments, (see Chapter 6).

### 2.4.5 Text Editor (Option 62)

TEXT EDIT | ESC X | 1BH, 58H

The TEXT EDIT function causes the text editor program to be brought in from the disk and to be executed with the current device assignments, (see Chapter 6).

### 2.4.6 Z-80 Assembler (Option 63)

ASMB | ESC A | 1BG, 41H

The ASMB function causes the Z-80 assembler to be brought in from the disk and to be executed using the current device assignments, (see Chapter 6).

### 2.4.7 PROM Programmer (Option 52)

PROM PGMR | ESC P | 1BH, 5ØH

The PROM PGMR function causes the PROM Programmer main program to be loaded and executed with the current device assignments, (see Chapter 6).

### 2.4.8 Jump to User Function

F1 | F2 | F3 | F4 | F5 | F6 | F7 | F8 |

ESC J digit7 | 1BH, 4AH, hexdigit7

where

digit7 ::= Ø | 1 | 2 | 3 | 4 | 5 | 6 | 7

hexdigit7 ::= 3ØH | 31H | 32H | 33H | 34H | 35H | 36H | 37H

This command causes control to be given to the user defined function loaded at the address specified in a RAM jump table. Note that the function F1 is equivalent to ESC J Ø. The jump table may be set up using CPUOS, and the user functions may be defined with the aid of the Z-80 assembler or BASIC. This group of commands will not be discussed further in this manual.

# 3. CATHODE RAY TERMINAL OPERATING SYSTEM (CRTOS)

The CRTOS operating program is the standard software provided with the basic system. It enables the Chromatics CG terminals to act as normal intelligent terminals, but with full color, high-resolution graphics. CRTOS is also used to enter the other (optional) operating programs. The logical device assignment tables for these other programs are normally established with CRTOS.

Throughout this chapter, illustrative key sequences will be given to demonstrate the various features using the conventions of section 1.4. Comments may be shown, (usually in lower case), to the right of the key sequence. Remember that a key sequence may be spread over several lines for convenience and readability, but carriage returns and line feeds will always be explicitly indicated by RETURN and LF, respectively. The reader will benefit by trying out each key sequence as it occurs.

## 3.1 Entry into CRTOS

There are two escape code functions for entering CRTOS, (CRTOS may also be entered using the RESET key, see section 2.4.1). The action of these functions is disucssed in detail below.

### 3.1.1 BOOT

| BOOT | ESC G | 1BH, 47H |
|---|---|---|

The BOOT function is used to clear the system and restart with a known state. The BOOT function initializes memory, (with the exception of the

buffer memory, Option 73), executes an erase page and finally executes the CRTOS function described below. Part of the memory initialized sets up the four windows. Each window has the following initial conditions:

- window size: full screen
- colors: foreground= white, background= black
- cursor: white, blind, in home position
- character mode, horizontal, character size 1 by 1
- decimal coordinate input
- A7 off, BLINK off, ROLL off, PLOT off, FILL off, BACKGROUND off
- OVERSTRIKE off

except that Window #0 has a visible cursor. Striking the BOOT key should result in a black screen with two blinking white lines, (the cursor), in the upper left hand corner.

### 3.1.2 CRTOS

| | | |
|---|---|---|
| CRTOS | ESC T | 1BH, 54H |

The CRTOS function causes the CRTOS operating program to begin execution; device assignments unchanged.

## 3.2 Entry into Other Programs

CRTOS is the standard entry point into the rest of the system. All of the escape codes mentioned in section 2.4 may be used to enter directly into any of the other system software.

## 3.3 Other Escape Functions

The following functions properly belong to the escape code processing section, however they are presented here since they primarily affect the CRTOS operating program.

### 3.3.1 Set Communications Mode

When operating as a terminal, the system may be in one of three modes: Local, Half duplex or Full duplex. The mode is set by one of the three following key sequences:

| | | |
|---|---|---|
| ESC L | 1BH, 4CH | Local mode |
| ESC H | 1BH, 48H | Half duplex mode |
| ESC F | 1BH, 46H | Full duplex mode |

In Local mode (Figure 3.1) characters are passed directly from logical input device AI to logical output device AO. This is normally used to send characters from the keyboard to a window, (CRT screen). Half duplex (Figure 3.2) and Full duplex (Figure 3.3) are designed for typical terminal communication with a remote computer. Characters are routed from AI to BO and BI to AO. In Half duplex mode, characters from AI are also sent to AO. Half duplex should be used if the host computer does not "echo" each character sent, otherwise, Full duplex should be used.

### 3.3.2 Set Communications Rate

ESC R sio ratecode | 1BH, 52H, hsio, hratecode

where

sio ::= Ø | 1

ratecode ::= Ø | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F

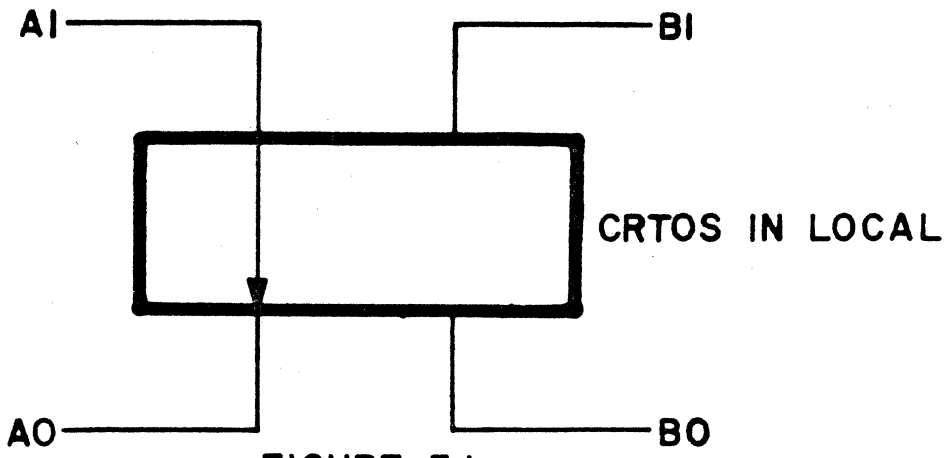



FIGURE 3.1

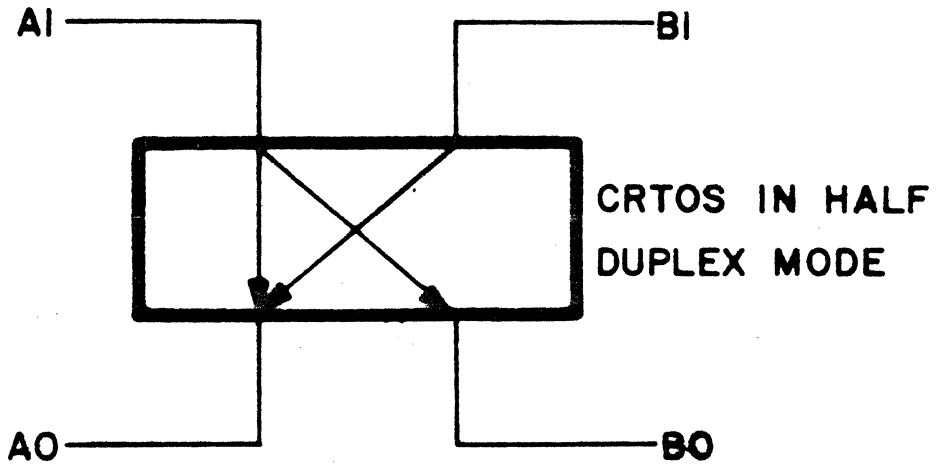



FIGURE 3.2

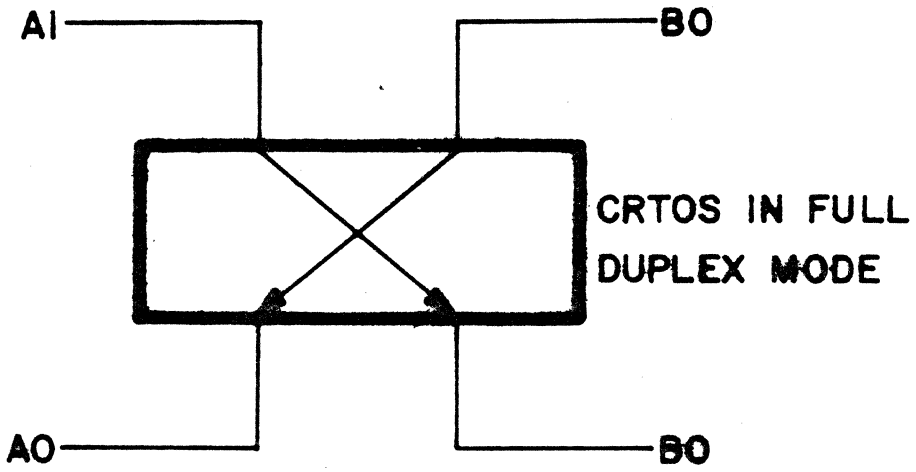



FIGURE 3.3

hsio ::= 3ØH | 31H

hratecode ::= 3ØH | 31H | 32H | ··· | 45H | 46H

The hexadecimal codes correspond left to right with the key codes in meaning. The value of sio indicates which Serial I/O port is having its communications rate set, (sio=Ø references SIO #Ø). The correspondence between the ratecode and BAUD rate set is given in the table:

Table 3.1

| ratecode | BAUD rate | ratecode | BAUD rate |
|---|---|---|---|
| Ø | 5Ø | 8 | 18ØØ |
| 1 | 75 | 9 | 24ØØ |
| 2 | 11Ø | A | 36ØØ |
| 3 | 15Ø | B | 48ØØ |
| 4 | 3ØØ | C | 96ØØ |
| 5 | 6ØØ | D | 125ØØ |
| 6 | 9ØØ | E | 25ØØØ |
| 7 | 12ØØ | F | 3125Ø |

Note: when BOOT is executed, SIO #0 is pre-assigned to 9600 BAUD and SIO #1 is pre-assigned to 11Ø BAUD.

### 3.3.3 Set Parity and Stop Bits

ESC S sio pscode | 1BH, 53H, hsio, hpscode

where

sio ::= Ø | 1

pscode ::= Ø | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8

hsio ::= 30H | 31H

hpscode ::= 30H | 31H | ··· | 38H

This command sets the parity (none, odd or even) and the number of stop bits (1, 1.5 or 2) used for asynchronous input and output on the indicated SIO port. The default values are no parity and one stop bit. The value of pscode to use to get the various other combinations can be determined from the following table:

| | Parity: | none | odd | even |
|---|---|---|---|---|
| 1 | Stop Bit | Ø | 3 | 6 |
| 1.5 | Stop Bits | 1 | 4 | 7 |
| 2 | Stop Bits | 2 | 5 | 8 |

### 3.3.4 Transmit Cursor Position

ESC Y window | 1BH, 59H, hwindow

where

window ::= Ø | 1 | 2 | 3

hwindow ::= 3ØH | 31H | 32H | 33H

The X and Y coordinates (see section 3.5) are transmitted to logical device BO, (normally assigned to an I/O port but could be assigned to another window).

### 3.3.5 The BREAK Key

BREAK

The BREAK key is special in that the keyboard handler does not generate a seven bit ASCII code for it. Instead, it is passed on in its original form of 8ØH. This code is treated as a NULL and no action is taken for all system components except the SIO ports. When the BREAK code reaches an output SIO Port, a 200 msec. break is created on the output line. This momentary line drop is used for special purposes on many communication systems.

### 3.3.6 Send ESC

ESC ESC

The Chromatics escape code processor traps all ESC characters for use in Chromatics escape codes. The above function causes an ESC to be sent to the host computer when operating in full or half duplex. Specifically, this function sends a single ESC character to logical device AO in Local mode, BO in Full duplex and AO and BO in Half Duplex. (Note that the second ESC deactivates internal Escape code processing. See Fig. 2.3.)

### 3.3.7 Logical Device Assignment

ESC io logical physical | 1BH, hio, hlog, hphys

where

io ::= I | O

logical ::= A | B | C | D | E

physical ::= Ø | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

hio ::= 49H | 4FH

hlog ::= 41H | 42H | 43H | 44H | 45H

hphys ::= 3ØH | 31H | 32H | ··· | 38H | 39H

The value of io determines whether a logical input (I) or output (O) device is to be assigned. For example, ESC I A Ø assigns logical input device AI to physical device Ø, while ESC O D 6 assigns logical output device DO to physical device 6. The physical devices are coded as follows:

Table 3.2

| physical | device |
|---|---|
| Ø | Window #Ø |
| 1 | Window #1 |
| 2 | Window #2 |

| physical | device | |
|---|---|---|
| 3 | Window #3 | |
| 4 | Serial I/O port #Ø (SIO #Ø) | |
| 5 | Serial I/O port #1 (SIO #1) | (Option 31) |
| 6 | Parallel I/O port #Ø (PIO #Ø) | (Option 33) |
| 7 | Parallel I/O port #1 (PIO #1) | (Option 33) |
| 8 | IEEE-488 (GPIB) | (Option 35) |
| 9 | Keyboard | |
| A-F | Unassigned (dummy) | |

Note: Each logical device may have two physical devices assigned to it. One of these is fixed, (see section 2.2). The above command changes the variably assigned physical devices.

## 3.4 The CRT Display Screen

The remainder of this chapter is concerned with generating displays on the CRT screen. Local mode is used throughout. Information is sent to the refresh memory, (and thereby displayed on the CRT screen), by sending it to a logical output device which is assigned to one of the four windows. For simplicity, only Window #Ø, (which is initially assigned to logical device A0), will be used until section 3.8, which will discuss the use of multiple windows.

A window is logically a rectangular subset of the points displayed on the CRT screen. Up until section 3.8, a window will be used at its default size, equal to the full screen. For clarity, the screen size will be assumed to be 512 x 256, (models CG 1398, 1598 and 1998). Appropriate comments relating to the 512 x 512 screen, (models CG 1399, 1599 and 1999), will be given in square brackets: [ ]

## 3.5 Coordinate Entry

Each point on the CRT screen is individually addressable by a pair of integers called coordinates. The first element of the pair is the X coordinate, and the second is the Y coordinate. The origin, (point (Ø,Ø) ), is located at the bottom left hand corner of the screen. The X coordinate gives the horizontal displacement in points to the right of the origin, and the Y coordinate gives the vertical displacement up from the origin. The value of X ranges from Ø to 511. The value of Y ranges from Ø to ymax where ymax = 255 [or 511].

### 3.5.1 Decimal Coordinate Mode

When working from the keyboard, coordinates are most conveniently entered in decimal mode, to be discussed in this subsection. The window is placed in decimal mode when BOOT is executed, but it may be re-entered with the following command:

MODE E | Ø1H, 45H

While in decimal mode, numbers (used in various functions) are entered as defined below:

number ::= digit digit digit | digit digit , |
digit ,

digit ::= Ø | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

That is, a number is either a string of three decimal digits, or one or two digits followed by a comma. In the hexadecimal coding of functions, numbers are represented as direct encodings of the above:

hnumber ::= hdigit hdigit hdigit | hdigit hdigit 2CH |
hdigit 2CH

hdigit ::= 3ØH | 31H | 32H | ··· | 39H

If an error is made in keying a number, a minus sign (-) may be used for correction. The minus sign effectively erases the partially entered number and allows it to be re-entered.

Screen coordinates, (the locations of particular points), are entered as pairs of numbers. It is sometimes convenient to use the cursor to specify coordinates, rather than entering them numerically. The cursor must be moved to the desired location on the screen, (see section 3.6), and the period (.) character entered. The coordinates of the cursor are then used as if they had been entered as a pair of numbers. Note: if a coordinate has been partially entered when the period is entered, the cursor coordinate supplants the partially entered numerical values. In the rest of the chapter, coordinates will be indicated by coord and hcoord:

coord ::= number number | .

hcoord ::= hnumber hnumber | 2EH

### 3.5.2 Binary Coordinate Mode

MODE B | ØlH, 42H

The above command causes entry into binary coordinate mode. Decimal coordinate mode may be re-entered as shown in 3.5.1. In binary mode, each number is entered as two bytes with the following formats:

| | Bit Positions | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | Ø |
| First byte | X | 1 | A5 | A4 | A3 | A2 | A1 | AØ |
| Second byte | X | 1 | X | X | X | A8 | A7 | A6 |

The value of the number is interpreted as a binary quantity with bits: A8 A7 A6 A5 A4 A3 A2 A1 AØ. Binary mode is intended for use in sending numbers for the various functions from another computer; only two characters rather than three need to be sent, and it is unnecessary to convert from internal binary form to decimal. However, binary form is very inconvenient to use from the keyboard and will not be discussed further in this manual.

## 3.6 Cursor Control

The cursor is a movable reference point within the window. In character mode, it is represented by two horizontal blinking lines one standard character width (6 or 8 dots) long displayed on the screen one above the other. The vertical displacement of the lines depends on the currently defined height of a character. The top line coincides with the top row of dots of the current character position, while the bottom line coincides with the bottom row of dots. The coordinate location of the character mode cursor is defined to be in the location of the leftmost dot of the top line.

The plot mode cursor is represented by a single dot followed by a horizontal line, both of which are blinking. The total width of the dot and line is equal to the width of a standard character. The coordinate location of the plot mode cursor corresponds to that of the dot.

### 3.6.1 Cursor Display Control

Occasionally, it is useful to make the cursor invisible on the screen. The visibility of the cursor and its color are controlled by the following commands.

3.6.1.1 Visible Cursor

MODE J | ØlH, 4AH

The cursor is made visible and blinking on the CRT screen.

3.6.1.2 Blind Cursor

MODE K | ØlH, 4BH

The cursor is made invisible on the CRT screen; however, it still exists and will be moved by the appropriate commands.

3.6.1.3 Change Cursor Color

MODE Q colnum | ØlH, 5lH, hcolnum

where

colnum ::= Ø | 1 | 2 | 3 | 4 | 5 | 6 | 7

hcolnum ::= 3ØH | 31H | 32H | 33H | 34H | 35H | 36H | 37H

The cursor color is changed to that indicated by the following table:

Table 3.3

| colnum | color | colnum | color |
|---|---|---|---|
| Ø | BLACK | 4 | RED |
| 1 | BLUE | 5 | MAGENTA |
| 2 | GREEN | 6 | YELLOW |
| 3 | CYAN | 7 | WHITE |

3.6.2 Character Related Cursor Movements

The following commands move the cursor integral numbers of character positions without altering the information displayed on the CRT screen. (see section 3.7 for information on the dimensions of a character position). These commands work for both character and plot modes. An important feature of the movements is that they will never take the cursor outside of the window.

Characters may start at any dot position within a window. For each possible character position, there is a "next" character position for each of four directions: left, right, up and down. This position is normally the adjacent character-sized rectangle of dots in the indicated direction; however, if this movement would carry the cursor outside the window, the software moves the cursor to the position indicated in sections 3.6.2.1 - 3.6.2.7.

A horizontal row of characters is called a "line". The cursor position defines the current line. The "first character position" of a line always coincides with the leftmost edge of the window. The "last character position" of a line is the last position of the line which is an integral number of positions from the first character position and is still within the window. Note that there may be excess points between the last character position and the edge of the window, (since windows are not necessarily even multiples of characters in size). The "top line" and "bottom line" of a window are defined analogously to the first and last characters of a line. With these definitions, the character related cursor movements can now be defined.

3.6.2.1 Home

HOME | CTRL | 1CH

The cursor is moved to the first character position of the top line.

3.6.2.2 Carriage Return

RETURN | CTRL M | ØDH

The cursor is moved to the first character position of the current line. Note that, unlike a typewriter, the RETURN function does not cause indexing to the next line.

3.6.2.3 Tabulation

TAB | CTRL I | Ø9H

The cursor is moved to the next character position on the current line which has a number divisible by eight, where the numbers are assigned left to right beginning with the first character position. If this action would move the cursor outside the window, the cursor is moved to the first character position on the next line; if the next line would be outside the window, the cursor is moved to the Home position.

3.6.2.4 Line Feed

↓ | LF | CTRL J | ØAH

The cursor is moved to the position immediately below the current one. If this would cause the cursor to go outside the window, the cursor is moved to the corresponding location on the top line.

3.6.2.5 Backspace

← | CTRL H | Ø8H

The cursor is moved to the position immediately to the left of the current position. If this goes outside the window, then the cursor is moved to the last position of the previous line; if the previous line would be outside the window, then the cursor is moved to the last position on the bottom line.

3.6.2.6 Vertical Tabulation

↑ | CTRL K | ØBH

The cursor is moved to the position immediately above the current position. If this is outside the window, then the cursor is moved to the corresponding position on the bottom line.

3.6.2.7 Cursor Right

→ | CTRL ] | 1DH

The cursor is moved to the next character position to the right of the current position. If this position is outside the window, the cursor is moved to the first character position of the next line; if the next line would be outside the window, the cursor is moved to the Home position.

3.6.2.8 Set Interline Spacing

MODE A number | ØlH, 41H, hnumber

Normally, each line of characters is directly adjacent to the line of characters above it. This corresponds to the default value of zero for interline spacing. The above command sets the number of points between lines of characters to number, (range Ø to 255). Note that the line of points between character lines are not written, (set to foreground or background color), when characters are sent with non-zero interline spacing. An erase page command, however, will always set all window lines to background color.

3.6.3 Point Related Cursor Movements

It is sometimes useful, (especially in plot mode), to be able to position the cursor to particular points within the window. The first four commands below allow the cursor to be moved one dot position at a time. With these commands, the coordinates of the cursor, (the upper left hand corner of the cursor position), will be kept within the window, but part of the current character position may extend beyond the window. This could lead to unexpected or undesired results if not used carefully.

Note that dot-by-dot movement of the cursor can position it at points which are not an integral number of character positions from the edges of the window. This is the reason that the movements in subsection 3.6.2 were defined in terms of the current character position rather than absolute character positions.

3.6.3.1 Cursor One Dot Up

SHIFT ↑ | CTRL E | Ø5H

The cursor is moved one dot position up. If this would be outside the window, the cursor is positioned to the bottom point within the window directly below the current position.

3.6.3.2 Cursor One Dot Down

SHIFT ↓ | CTRL V | 16H

The cursor is moved one dot position down from the current position. If this would move outside the window, the cursor is positioned to the top point within the window directly above the current cursor position. (See 3.6.2.4)

3.6.3.3 Cursor One Dot Left

SHIFT ← | CTRL Y | 19H

The cursor is moved one dot position to the left. If this would be outside the window, the cursor is positioned at the rightmost point of the line of points one point above the current position.

3.6.3.4 Cursor One Dot Right

SHIFT → | CTRL_ | 1FH

The cursor is moved one dot position to the right. If this is outside the window, then the cursor is positioned to the leftmost point on the line of points immediately below the current position.

3.6.3.5 Cursor to Coordinate

CURSOR X-Y | MODE U coord | ØlH, 55H, hcoord

The cursor is moved directly to the coordinates indicated. This command ignores window boundaries and should be used with care. (See subsection 3.5.1 for an explanation of coord and hcoord).

## 3.7 Character Mode

The terminal is automatically placed in character mode when CRTOS is entered. Any of the non-control ASCII characters will be displayed on the screen in the current character position when the appropriate key or keys are struck. In the default character size, (1 x 1), a character position is a 6 dot wide by 1Ø dot high matrix. (An 8 dot by 1Ø dot character position is available with Option 27). Graphics characters use all 6Ø dots, but normal alphanumeric characters use only 5 by 7 dots. The remaining dots give intercharacter and interline spacing. The following example illustrates character mode use of the terminal, (the results should be similar to the comment area):

```
BOOT C H R O M A T I C S SPACE C G RETURN LF        CHROMATICS CG
LF
SHIFT 4 SHIFT A                                     $a
```

### 3.7.1 Alternate Character Sets

Any of four display character sets of 96 characters each can easily be displayed on the terminal. The alternate character sets can be reached using the A7 bit, which is set using the following commands.

3.7.1.1 A7 On

CTRL N | ØEH

This command turns on the A7 bit. All characters are taken from the alternate character set, (normally the set defined in Figure 1.3), while this bit is on.

3.7.1.2 A7 Off

CTRL O | ØFH

The command turns the A7 bit off, (the normal condition), and causes all characters to be taken from the standard ASCII set.

3.7.1.3 Select Upper Character Set Mode

MODE S cset | Ø1H, 53H, hcset

where

cset ::= Ø | 1 | 2 | 3

hcset ::= 30H | 31H | 32H | 33H

This command selects which character set is to be used when the A7 bit is on. If cset = Ø, (the default condition), then the upper character set corresponds to the graphics set defined in Figure 1.3. If Option 21 is purchased, then cset = 1 or 2 refers to user defined special characters in PROM. A value of 3 for cset causes the upper character set to be taken from RAM, (see section 5.5.3).

3.7.2 Set Character Size

The normal character position is 6 horizontal points by 10 vertical points. (With Option 27, a character position is 8 by 10. A size of 6 by 10 is assumed throughout this manual). The character size can be expanded an integer number of times in either the horizontal or vertical directions using the following commands. The characters displayed are automatically expanded to fit the new character positions.

3.7.2.1 Set Character Height

MODE Y number | ØlH, 59H, hnumber

The vertical dimension of a character is set to number times the standard character size. The maximum value of number is 25 [51]. See subsection 3.5.1 for a description of number.

3.7.2.2 Set Character Width

MODE X number | ØlH, 58H, hnumber

The horizontal dimension of a character is set to number times the standard character width. The maximum value of number is 85.

Example key sequence:

```
BOOT A B C RETURN LF LF
MODE Y 3, A B C RETURN LF LF
MODE X 3, A B C
```

When keyed correctly, this sequence should produce three lines with "ABC" displayed, the first line normal sized, the second tall and narrow and the third, normal characters three times larger, (in both dimensions.)

3.7.3 Character Input Movement

Normally, when a character is input it is placed at the current cursor location and the cursor is moved as if a cursor right command had been given, so that characters are displayed horizontally. Occasionally it may be convenient to display the characters vertically down the page. This is controlled by the following pair of commands.

3.7.3.1 Write Horizontal Mode

MODE H | Ø1H, 48H

Successive characters are written horizontally across the window from left to right. This is the default mode.

3.7.3.2 Write Vertical Mode

MODE V | Ø1H, 58H

Successive characters are written vertically down the window from top to bottom. This is identical to writing the character in the current cursor position and moving the cursor with a line feed, (3.6.2.4), rather than a cursor right. Note: All cursor movements work exactly as in horizontal mode.

Example key sequence: Should result in:

BOOT A B C MODE V A B C A B C

3.7.4 Selecting Character Color and Blinking

The mode of the window may be set to a foreground color, a background color and foreground and/or background blinking. The form in which characters are displayed depends on the mode of the window at the time the characters are entered. The character is set to the foreground color, while the remaining points within its character position are set to the background color. If foreground blink is on, the character will blink at 1.9 Hz. If background blink is on, the background color will blink at a 1.9 Hz rate. The colors and blinking are controlled by the following commands.

3.7.4.1 Select Color

BLACK | BLUE | GREEN | CYAN | RED | MAGENTA | YELLOW | WHITE |

MODE C digit7 | ØlH, 43H, hdigit7

This command sets either the foreground or the background color depending on whether background mode is on or off. (Note: digit7 and hdigit7 are defined in subsection 2.4.8 MODE C Ø corresponds to BLACK and MODE C 7 corresponds to WHITE.)

3.7.4.2 Set Background On

BACKGROUND | MODE M | ØlH, 4DH

This command sets background mode on so that future color commands will affect the background color. The BACKGROUND key is lighted.

3.7.4.3 Set Background Off

BACKGROUND* | MODE N | ØlH, 4EH

This command sets background mode off, (the default condition), so that future color commands will affect the foreground color. The BACKGROUND key light is turned off. (Note: the first form of this command indicates that the BACKGROUND key must be lit for this key to cause the desired action).

3.7.4.4 Blink On

BLINK | MODE 1 | ØlH, 31H

This command sets the window mode to either foreground blink or background blink. Which is set depends on whether the background off or background on command has been most recently given. See the example after 3.7.4.5. (Note: If the blink key is lit when BACKGROUND or BACKGROUND* is entered, then a blink on is automatically generated; similarly if the blink key is not lit, a blink off is generated).

3.7.4.5 Blink Off

BLINK* | MODE 2 | ØlH, 32H

This command sets the window mode to either foreground or background non-blink, depending on whether background mode is off or on. The following example should be helpful:

```
BOOT MODE X 5,  MODE Y 5,  BACKGROUND BLUE ERASE PAGE
CYAN  BACKGROUND* RED  → → ↓ ↓
F  BLINK  →  F  BACKGROUND  →  F  BACKGROUND* BLINK* →  F
```

When entered correctly, all four possibilities for blinking should be displayed.

3.7.5 Additional Display Functions

The following three commands complete the character related display functions for the terminals. Of these, the first is by far the most used.

3.7.5.1 Erase Page

ERASE PAGE | CTRL L | ØCH

This command causes the entire window to be cleared to the current background color and the cursor to be positioned to the home position. This command is sometimes also called a form feed.

3.7.5.2 Erase Line

MODE @ | ØlH, 4ØH

This command causes the current line to be cleared to the background color. The cursor is not moved.

3.7.5.3 Test

TEST char | MODE T char | ØlH, hchar

where

char ::= any displayable character

hchar ::= 2ØH | 21H | ··· | 7FH

This command causes an erase page followed by the filling of the window with the indicated character, (which cannot be a control character).

3.7.5.4 Erase to End of Line

MODE 3 | Ø1H, 33H

This command causes the remainder of the current line from the cursor to the right hand edge of the window to be erased to background color. The cursor remains at its original location.

3.8 Multiple Windows

In the preceding sections, only a single window, (Window #Ø), has been used. The system actually supports four windows, (Windows #Ø, #1, #2 and #3). All of the functions presented in section 3.5, 3.6 and 3.7 can be individually sent to each window. The sizes of the windows can be set with the following command.

3.8.1 Set Window Size

WINDOW coord coord | MODE W coord coord |

Ø1H, 57H, hcoord, hcoord

The pair of coordinates in the command define a rectangular subset of the CRT screen to be the points of the window. See section 3.5 for information on coord and hcoord. The following key sequence defines Window #Ø to be the upper left quadrant of the screen and fills the window with Q's:

```
BOOT WINDOW Ø, 255 255 127 TEST Q
[BOOT WINDOW Ø, 511 255 255 TEST Q]
```

### 3.8.2 Addressing Multiple Windows

So far, only Window #Ø has been used because it is the window assigned to output A0 by default. In order to affect the other windows, they must be assigned to the appropriate logical output devices. This is normally done by reassigning logical output device A0. It should be noted that assigning a window does not affect the status of the lights in the illuminated keys; therefore, the lights on the keys do not indicate the mode of the window, but instead reflect the meaning of the key. An example should help to clarify the situation.

Example:

```
BOOT WINDOW Ø, 255 255 127          window #Ø = upper left quad.
BACKGROUND GREEN  ERASE PAGE        make window visible
ESC 0 A 1                           assign A0 to window #1
WINDOW  256 255 511 127             window #1 = upper right quad.
MODE J                              make cursor visible
RED  A B C
```

Note that although the BACKGROUND key is illuminated, the final line does not set the background color to red, but rather the foreground color. Striking the BACKGROUND key will realign it with the mode of the window. To change the background color of Window #1, the BACKGROUND key must be struck twice. Alternately, the direct command of MODE M may be given as in the continuation of the above example.

Continued example:

```
MODE M RED D E F ERASE PAGE         background of w #1 = red
```

The background color of Window #1 has now been set to red, as desired. (Note that red-on-red characters are not terribly legible).

Continued example:

```
BACKGROUND* BLACK                   foreground of w #1 = black
MODE X 3, MODE Y 3,                 char. size 3 x 3
A B C
```

Window #1 has now been set up with entirely different characteristics from Window #Ø. Window #Ø retains its identity as shown by the continuation of the example:

Continued example:

```
ESC O A Ø                          assign AO to window #Ø
A B C                              continue at last point
```

Difficulties with the window status compared with the illuminated keys can be avoided by using the following command whenever a new window is assigned to AO.

3.8.2.1 Keyboard sync

MODE Ø | Ø1H, 3ØH

This command causes the status of the addressed window to be sent to the keyboard illuminated keys. After a keyboard sync command, the window status will be accurately reflected by the illuminated keys.

3.8.3 Overlapping Windows

Since windows are simply defined as subsets of the CRT screen, there is nothing to prevent them from overlapping. In fact, under initial conditions, all of the windows coincide. When two windows share the same point on the screen, whichever window writes the point last is the one that affects that point. Consider the continuation of the previous example.

Continued example:

```
ESC O A 2                          assign AO to window #2
BACKGROUND MAGENTA ERASE PAGE      clear to magenta
ESC O A 1                          back to w #1
ERASE PAGE
```

3.8.4 Exceeding Window Boundaries

In subsection 3.6.3, the possibility of exceeding the window boundaries was mentioned. This will now be illustrated along with coordinate input using the cursor.

Example:

```
BOOT WINDOW . → → → → ↓ ↓ .          define w #Ø with cursor
BACKGROUND BLUE ERASE PAGE           make window visible
LF → SHIFT ↓ SHIFT ↓ SHIFT ↓ SHIFT ↓
A B C D E F G H I
ERASE PAGE
```

The window defined above is 25 points wide by 21 points high because the cursor coordinate is located at the upper left hand point within the character position. Note that the information written outside of the window is not cleared by the ERASE PAGE.

Window boundaries can also be exceeded if the window is too small to contain a single character. Continuing the above example:

```
MODE X 3, MODE Y 7, A B C
```

## 3.9 Graphic Functions

The Chromatics terminals provide fast, built-in functions for generating graphics figures. These functions are enabled when the window is in plot mode. While in plot mode, some of the escape, control and mode functions defined previously do not affect graphics generation, (for example, background color); however, all of these functions are allowed and do affect the window status. Also, display character codes which do not have a special meaning in plot mode are ignored.

### 3.9.1 Entering Plot Mode

```
PLOT | MODE G | ØlH, 47H
```

Plot mode is entered, (initially in the DOT submode - see 3.9.5), and all functions described in this section become active. The cursor is displayed as a blinking dot followed by a line.

### 3.9.2 Returning to Character Mode

PLOT* | CTRL U | 15H

Character mode is entered. This command is also treated as a "MODE Cancel"; any partially entered MODE command is ignored.

### 3.9.3 Coordinates in Plot Mode

Coordinates are entered in the manner described in section 3.5 Recall that, while using a window with a size less than the full screen, numeric coordinates may select points outside the window. This can result in graphic figures which extend beyond the window boundaries.

### 3.9.4 Dot Distances

Because the width and height of the CRT screen display area are not in the same ratio as the number of X dots and Y dots, the distance between a pair of X dots is not the same as the distance between a pair of Y dots. On the 512 by 512 screens, (CG 1399, 1599, 1999), the X dot distance is equal to the square root of two times the Y dot distance, (or, 5 X dots approximately equals 7 Y dots). On the 512 by 256 screens, (CG 1398, 1598, 1998), the X dot distance is equal to the inverse of the square root of two times the Y dot distance, (or, 7 X dots approximately equals 5 Y dots). To get dots per inch for a particular screen, divide the dimensions of the actual displayed area by the number of dots in the X and Y directions, respectively. (The actual displayed area may be made apparent by: BOOT BACKGROUND BLUE ERASE PAGE).

### 3.9.5 Plot Submodes

While in plot mode, the window is also in one of several submodes described below. The commands in this subsection are only available

# X BAR GRAPH

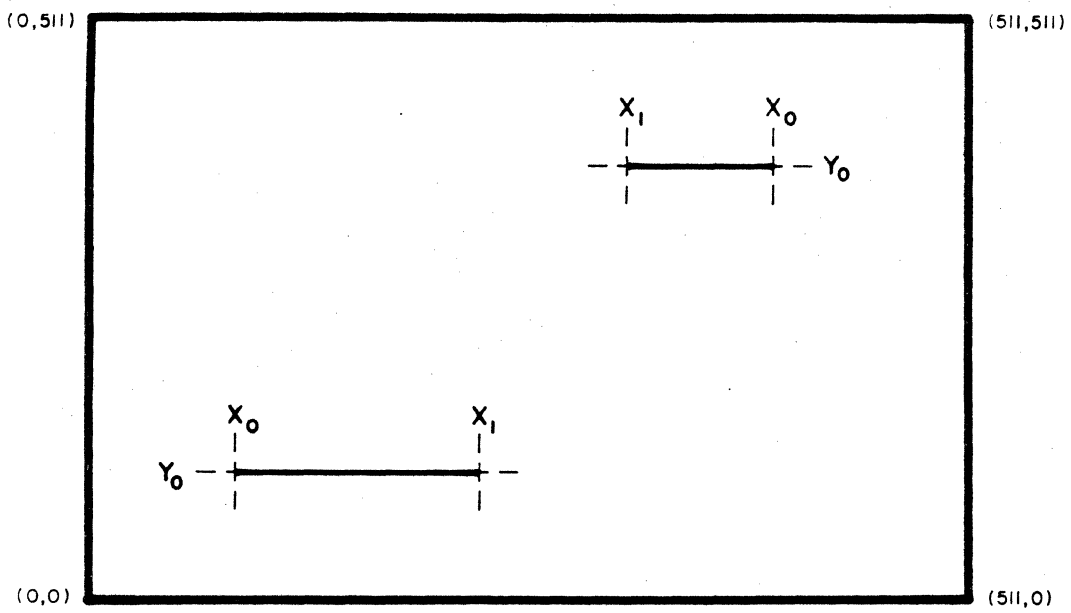



FIGURE 3.4

# Y BAR GRAPH

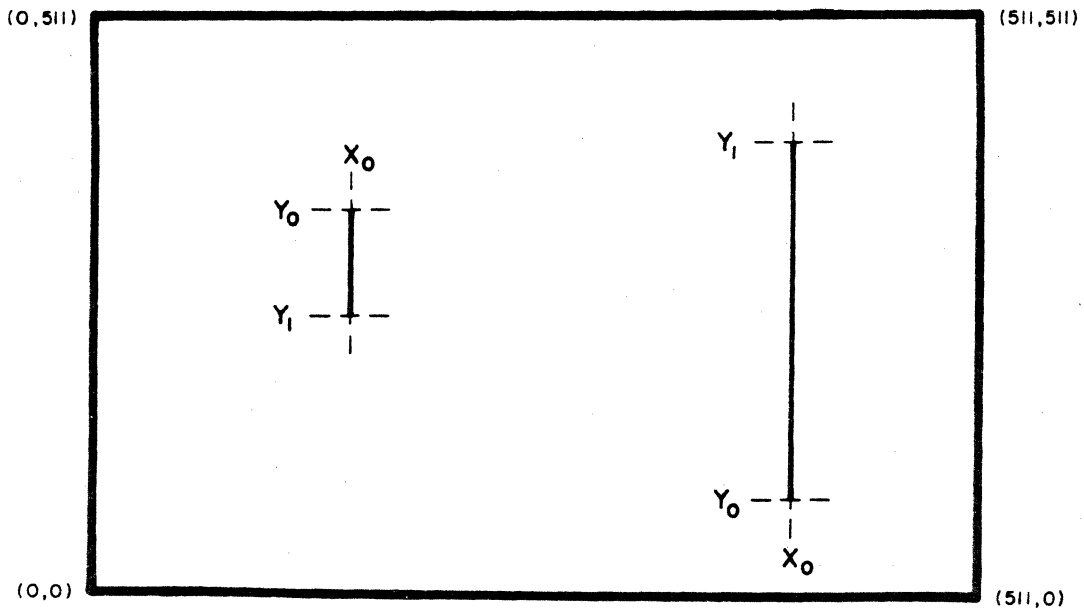



FIGURE 3.5

while in Plot mode. Each of these submodes causes subsequent number and/or coord input to be interpreted according to that submode. Only the initial entry is indicated in the command formats. Subsequent entries are indicated in the text. (Note: all examples in this section are assumed to follow the key sequence: BOOT PLOT).

3.9.5.1 X Bar Submode

X BAR coord xcoord | 21H, hcoord, xhcoord

where

xcoord ::= number

xhcoord ::= hnumber

This command places the window in the horizontal bar plot submode. The first and each subsequent coord xcoord pair causes a line to be drawn from coord horizontally to the point with X coordinate xcoord. If the cursor is used to enter the xcoord value, only the X coordinate of the cursor is used. The following example corresponds to Figure 3.4

```
X Bar  85,  45,  225   4ØØ   19Ø   31Ø
[X BAR 85,  90,  225   4ØØ   38Ø   31Ø]
```

3.9.5.2 Y Bar Submode

YBAR coord ycoord | 22H, hcoord, ycoord

where

ycoord ::= number

yhcoord ::= hnumber

This command places the window in vertical bar plot submode. The first and each subsequent coord ycoord pair causes a line to be drawn from coord vertically to the point with the Y coordinate ycoord. If the cursor is used to input ycoord, only the Y coordinate of the cursor is used. The following example corresponds to Figure 3.5.

```
Y BAR  14Ø  17Ø  127   4ØØ  Ø4Ø   2ØØ
Y BAR  14Ø  34Ø  255 . 4ØØ  Ø8Ø   4ØØ
```

3.9.5.3 Incremental X Bar Submode

# coord xcoord | 23H, hcoord, xhcoord

This command places the window in incremental horizontal plot submode. The first coord xcoord pair works the same as for X Bar. Subsequently, only xcoord values are entered. Bars are drawn as for X Bar where the coord value is taken as the previous coord value plus one in the Y direction. If the generated value of coord would be outside the window, the Y value is taken as that of the bottom of the window. The following example corresponds to Figure 3.6.

# INCREMENTAL X BAR

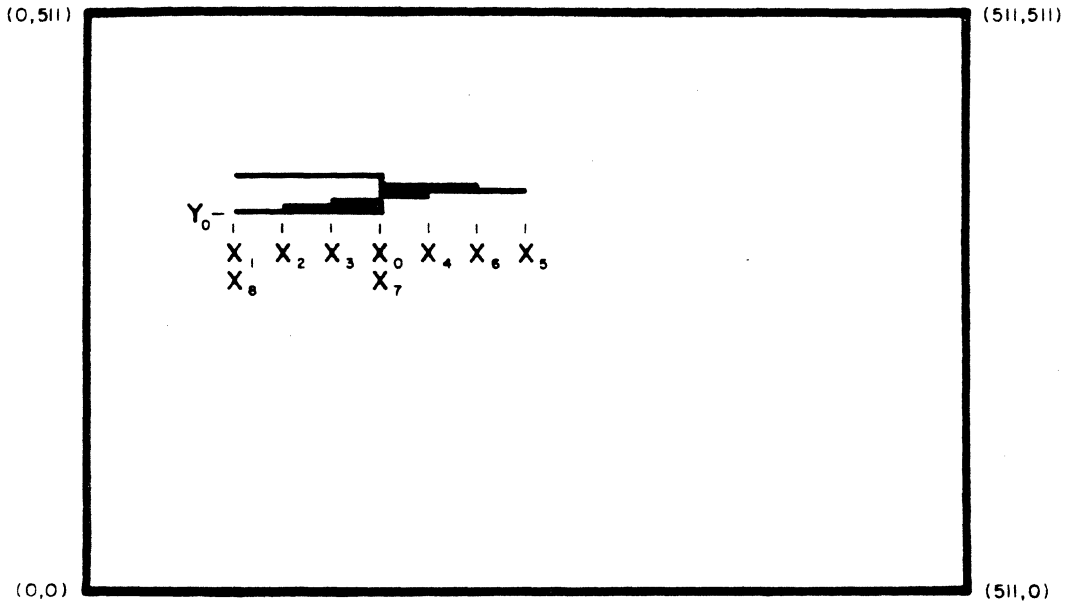



FIGURE 3.6

# INCREMENTAL Y BAR

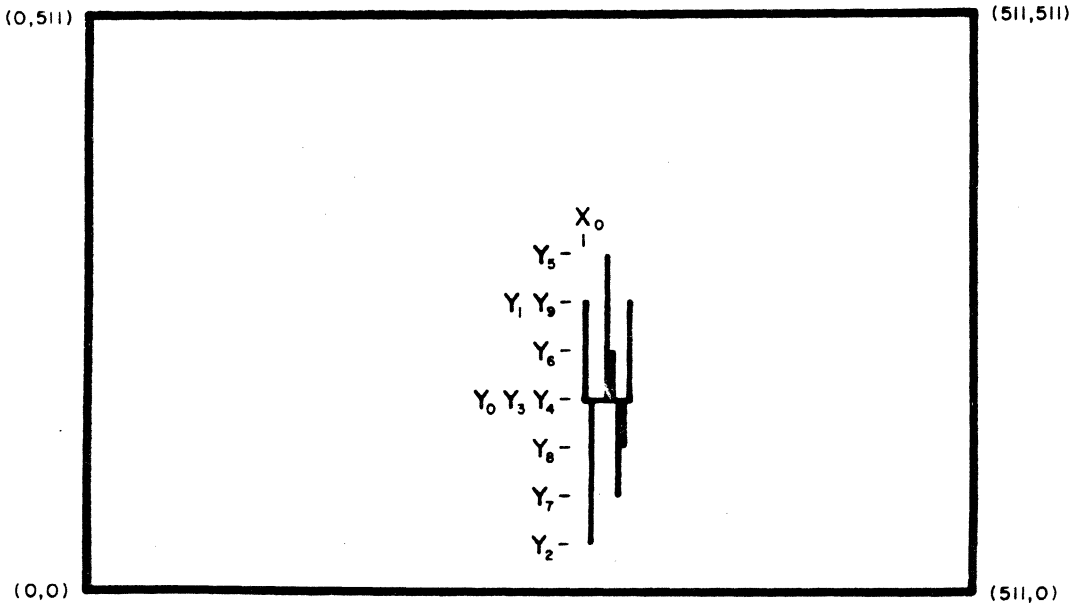



FIGURE 3.7

```
#  175  17Ø  85,  115  145  2Ø5  265  235  175  Ø85
[#  175  34Ø  85,  115  145  2Ø5  265  235  175  Ø85]
```

3.9.5.4 Incremental Y Bar Submode

$ coord ycoord | 24H, hcoord, yhcoord

This command places the window in incremental vertical plot submode. The first coord ycoord pair works the same as for Y Bar. Subsequently, only ycoord values are entered. Bars are drawn as for Y Bar where the coord value is taken as the previous coord value plus one in the X direction. If the generated value of coord would be outside the window, the X value is taken as that of the left edge of the window. The following example corresponds to Figure 3.7.

```
$  285  Ø85  127  2Ø,  85,  85,  15Ø  1Ø5  4Ø,  6Ø,  127
[$  285  17Ø  255  Ø4Ø  17Ø  17Ø  3ØØ  21Ø  Ø8Ø  12Ø  255]
```

3.9.5.5 Dot Submode

DOT coord | 25H, hcoord

This command places the window in the display dot plot submode. (This is the default submode when plot mode is entered). The first and each subsequent coord causes a dot to be displayed at the indicated point. The following example corresponds to Figure 3.8 (Note: the color change is made to make the display more visible).

```
WHITE BACKGROUND BLUE ERASE PAGE  DOT  4ØØ 17Ø 115 21Ø 23Ø Ø85
[WHITE BACKGROUND BLUE ERASE PAGE  DOT  4ØØ 34Ø 115 42Ø 23Ø 17Ø]
```

3.9.5.6 Incremental Dot Submode

& coord xydelta | 26H, hcoord, hxydelta

# X-Y DOT

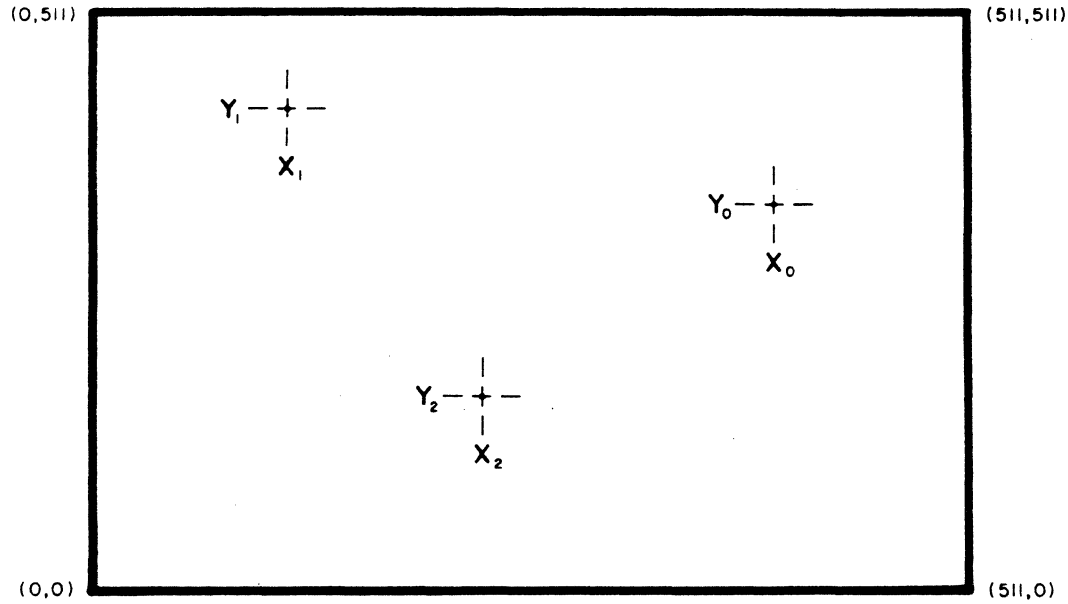



FIGURE 3.8

# INCREMENTAL X-Y DOT

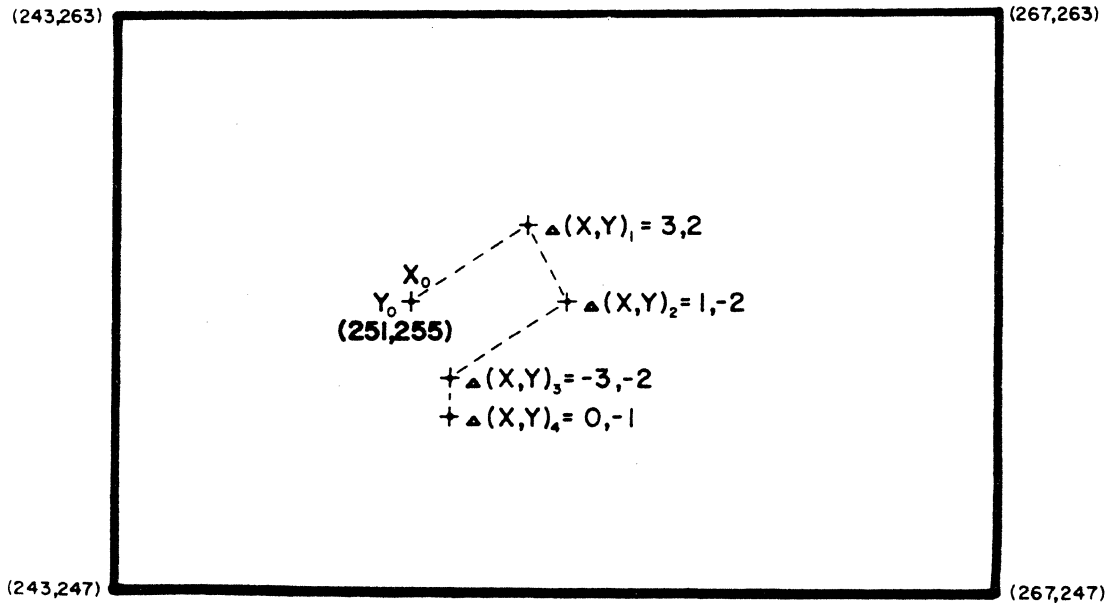



FIGURE 3.9

This command places the window in the incremental display dot plot submode. The coord indicates the placement of the first dot. The first and subsequent xydelta values indicate displacements of up to three dots in the X and/or Y directions. The displacement is added to the previous dot's coordinates, (taking window boundaries into account), to give the coordinates of the next dot to be written. Each xydelta byte is interpreted as follows, (independently of whether the window is in decimal or binary mode):

| | BIT POSITIONS | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | Ø |
| xydelta code: | _ | 1 | $X_s$ | $X_1$ | $X_Ø$ | $Y_s$ | $Y_1$ | $Y_Ø$ |

The binary values of $X_1X_Ø$ and $Y_1Y_Ø$ indicate the displacement magnitude. A value of Ø for $X_s$ or $Y_s$ indicates a direction of right or up, respectively, and a value of 1 for $X_s$ or $Y_s$ indicates a direction of left or down, respectively. The possible movements are summarized in Table 3.4 Also see Appendix A.5.

Table 3.4

## Summary of Incremental Dot Movements from @

| | | Dots Left | | | | Dots Right | | |
|---|---|---|---|---|---|---|---|---|
| | | 3 | 2 | 1 | | 1 | 2 | 3 |
| Dots Up | 3 | { | s | k | C | K | S | [ |
| | 2 | z | r | j | B | J | R | Z |
| | 1 | y | q | i | A | I | Q | Y |
| | | x | p | h | @ | H | P | X |
| Dots Down | 1 | } | u | m | E | M | U | ] |
| | 2 | ~ | v | n | F | N | V | ^ |
| | 3 | DEL | w | o | G | O | W | _ |

The following example corresponds to Figure 3.9. (The color change is made to make the diplay more visible.)

WHITE BACKGROUND BLUE ERASE PAGE & 255 127 Z N ∿ E

[WHITE BACKGROUND BLUE ERASE PAGE & 255 255 Z N ∿ E]

3.9.5.7 Vector Submode

VECTOR coord coord | 27H, hcoord, hcoord

This command places the window in the vector plot submode. The first and each subsequent pair of coords cause a line to be drawn between the two designated points. The following example corresponds to Figure 3.1Ø.

VECTOR 5Ø, 15Ø 145 225 115 Ø85 215 Ø85 345 21Ø 43Ø Ø65

[VECTOR 5Ø, 3ØØ 145 45Ø 115 17Ø 215 17Ø 345 42Ø 43Ø 13Ø]

3.9.5.8 Concatenated Vector Submode

( coord coord | 28H, hcoord, hcoord

This command places the window in the concatenated vector plot submode. The first pair of coords causes a line to be drawn as in vector mode. Each subsequent coord causes a line to be drawn from the end of the previous vector to the indicated point. The following example corresponds to Figure 3.11.

( Ø65 Ø65 145 1Ø5 2ØØ 165 375 Ø65 345 15Ø 46Ø 175 46Ø 225

[ ( Ø65 13Ø 145 21Ø 2ØØ 33Ø 375 13Ø 345 3ØØ 46Ø 35Ø 46Ø 45Ø ]

# RANDOM VECTORS

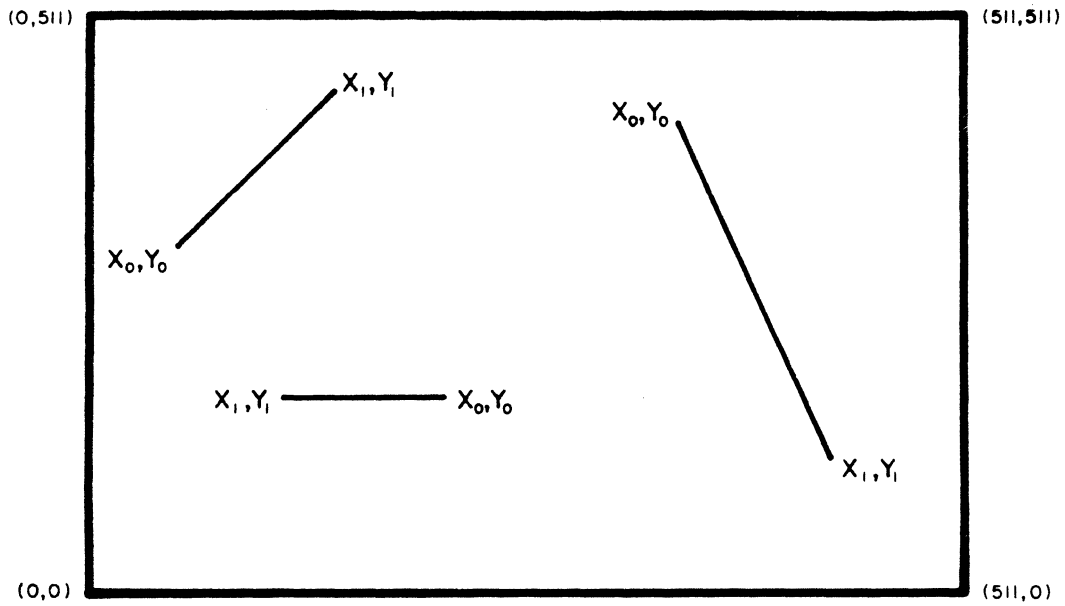



FIGURE 3.10

# CONCATENATED VECTORS

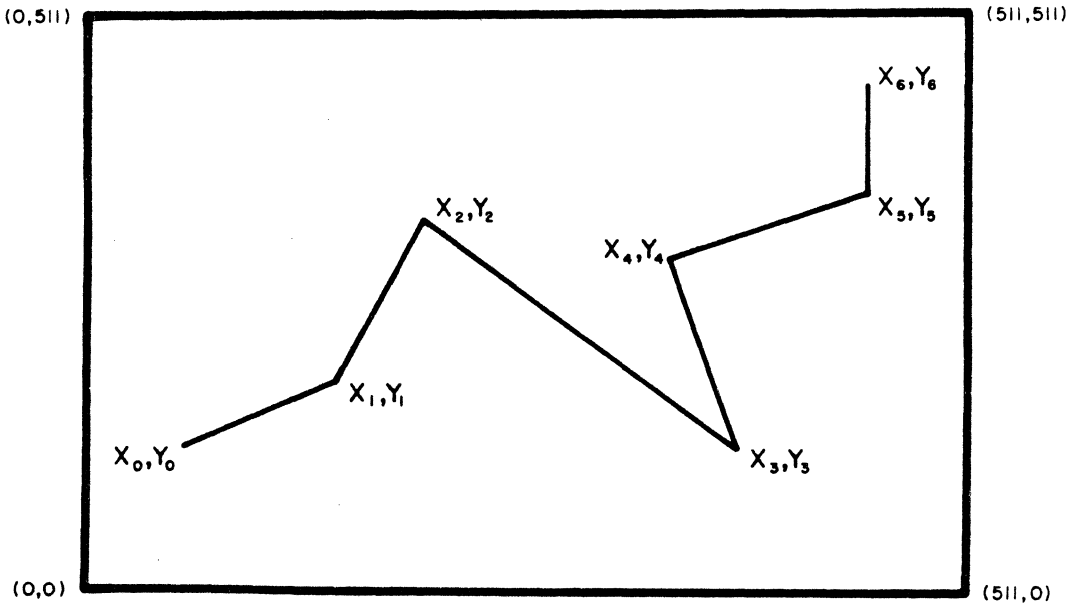



FIGURE 3.11

## 3.1Ø Summary of Standard Functions

This section summarizes the functions which are available in the standard system. For a complete summary of all functions, see Appendix A. The relationships between the various code types is shown in Figure 3.12.

### 3.1Ø.1 Control Functions

Control functions are specified by single, non-printing ASCII characters. Most control functions cause movements of the cursor. Two control functions, (CTRL [ and CTRL A), indicate that following characters are to be treated as modifiers for extended functions. These are discussed in subsequent subsections.

| Function | Name | Section | Function | Name | Section |
|---|---|---|---|---|---|
| CTRL A | MODE | 3.1Ø.3 | CTRL U | mode cancel | 3.9.2 |
| CTRL H | backspace | 3.6.2.5 | CTRL [ | ESC | 3.1Ø.2 |
| CTRL I | tab | 3.6.2.3 | CTRL \ | home | 3.6.2.1 |
| CTRL J | line feed | 3.6.2.4 | CTRL ] | curs. rt. | 3.6.2.7 |
| CTRL K | vert. tab | 3.6.2.6 | CTRL ^ | end file | 1.3 |
| CTRL L | erase pg. | 3.7.5.1 | CTRL E | dot down | 3.6.3.1 |
| CTRL M | return | 3.6.2.2 | CTRL V | dot left | 3.6.3.2 |
| CTRL N | A7 on | 3.7.1.1 | CTRL Y | dot up | 3.6.3.3 |
| CTRL O | A7 off | 3.7.2.2 | CTRL _ | dot right | 3.6.3.4 |

### 3.1Ø.2 Escape Functions

Escape functions are specified by the ESC character, (CTRL [), followed by a sequence of one or more characters. The first character after the ESC indicates the type of function. Only those escape functions which apply to the standard system are listed here. For more information on

# CODE FLOW-INPUT TO WINDOW

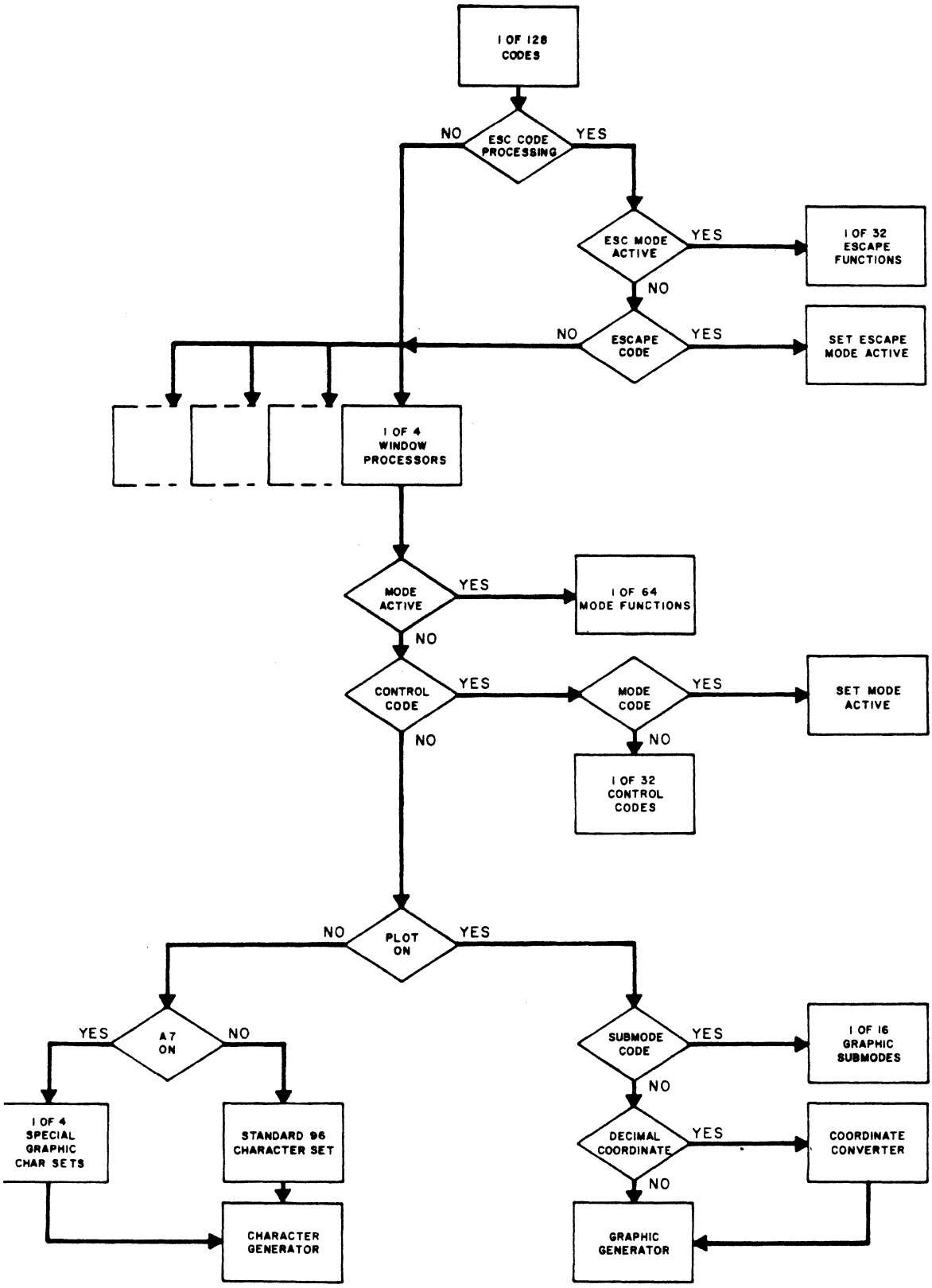



FIGURE 3.12

escape functions, see section 2.3.

| Function | Name | Section | Function | Name | Section |
|---|---|---|---|---|---|
| ESC F | full dupl. | 3.3.1 | ESC R | baud rate | 3.3.2 |
| ESC H | half dupl. | 3.3.1 | ESC I | asg. input | 3.3.7 |
| ESC L | local | 3.3.1 | ESC O | asg. output | 3.3.7 |
| ESC G | boot | 3.1.1 | ESC Y | trans. cur. | 3.3.4 |
| ESC T | CTROS | 3.1.2 | ESC ESC | send ESC | 3.3.6 |

### 3.1Ø.3 Mode Functions

The mode functions are used to establish the operating mode of a window. Subsequent commands sent to the window are affected by the mode. For example, the current color mode of the window defines the color with which characters will be written in that window.

| Function | Name | Section | Function | Name | Section |
|---|---|---|---|---|---|
| MODE @ | erase line | 3.7.5.2 | MODE S | sel. upper | 3.7.1.3 |
| MODE A | interline | 3.6.2.8 | MODE T | test | 3.7.5.3 |
| MODE B | binary co. | 3.5.2 | MODE U | CURSOR X-Y | 3.6.3.5 |
| MODE C | color | 3.7.4.1 | MODE V | vertical | 3.7.3.2 |
| MODE E | decimal | 3.5.1 | MODE W | window | 3.8.1 |
| MODE G | plot on | 3.9.1 | MODE X | X magnitude | 3.7.2.2 |
| MODE H | horizontal | 3.7.3.1 | MODE Y | Y magnitude | 3.7.2.1 |
| MODE J | visible cur. | 3.6.1.1 | MODE Ø | keybd. sync. | 3.8.2.1 |
| MODE K | blind cur. | 3.6.1.2 | MODE 1 | blink on | 3.7.4.4 |
| MODE M | back. on | 3.7.4.2 | MODE 2 | blink off | 3.7.4.5 |
| MODE N | back. off | 3.7.4.3 | MODE 3 | era. to EOL | 3.7.5.4 |
| MODE Q | cur. color | 3.6.1.3 | | | |

### 3.1Ø.4 Plot Submodes

When the window is in the plot mode, any of the plot submodes listed here can be entered.

| Submode | Submode Code | Section |
|---|---|---|
| X Bar | X BAR \| ! | 3.9.5.1 |
| Y Bar | Y BAR \| " | 3.9.5.2 |
| Incremental X Bar | # | 3.9.5.3 |
| Incremental Y Bar | $ | 3.9.5.4 |
| Dot | DOT \| % | 3.9.5.5 |
| Incremental Dot | & | 3.9.5.6 |
| Vector | VECTOR \| · | 3.9.5.7 |
| Concatenated Vector | ( | 3.9.5.8 |

# 4. EXTENDED DISPLAY FUNCTIONS

This chapter explains three optional features which may be added to the standard system, (in any combination), to further extend the already versatile alphanumeric and graphic capabilities.

## 4.1 Alphanumeric Mode Extension (Option 72)

This option provides the features described in this section for: scrolling, overstriking and inserting/deleting lines and characters.

### 4.1.1 Roll On

ROLL | MODE R | ØlH, 52H

This function places the window in roll mode. When in roll mode, the information in the window is automatically scrolled up and down as the cursor moves past the edge of the window at the bottom and top, respectively. More precisely, if the cursor is moved so that it would exceed the bottom boundary of the window, the entire contents of the window is shifted up by the X magnitude of one character, (or one raster line if the cursor dot moves are used) and an erase line is done on the bottom line of the window. The contents of the topmost line of the window are lost. Complementary actions take place if the cursor is moved beyond the top boundary of the window. (Note: when the window happens to be equal in size to the full screen, a very fast hardware roll is used and XY dot addresses will change position. If the window is less than full screen size, rolling is performed by somewhat slower software routines and the XY dot addresses remain stationary. The full screen hardware roll is particularly useful for high speed alphanumeric text applications.)

4.1.2 Roll Off (or Return to Page Mode)

ROLL* | MODE P | ØlH, 5ØH

The window is returned to the normal (page) mode.

4.1.3 Overstrike Character

MODE O | ØlH, 47H

The window is placed in overstrike mode for the next display character. When in overstrike mode, only the foreground dots of the character are written to the refresh memory - the background remains unchanged. This allows multiple characters to be overlapped in the same location. Consider the example:

BOOT MODE X 3, MODE Y 3, → → → ↓ ↓ ↓ BACKGROUND BLUE

8" SPACE / ← MODE O \ SPACE 1Ø"

4.1.4 Latch Overstrike

MODE ] | ØlH, 5DH

The window is placed in overstrike mode, (see 4.1.3) for all subsequent characters until overstrike mode is turned off. This function is useful in underlining as well as special applications. Consider the example:

BOOT MODE X 3, MODE Y 3, → → → ↓ ↓ ↓ BACKGROUND BLUE BACKGROUND*

C H R O M A T I C S RETURN → → →

RED MODE ] SHIFT CTRL / SHIFT CTRL / SHIFT CTRL /

SHIFT CTRL / SHIFT CTRL / SHIFT CTRL / SHIFT CTRL /

SHIFT CTRL / SHIFT CTRL / SHIFT CTRL / MODE [

### 4.1.5 Unlatch Overstrike

MODE [ | ØlH, 5BH

**The window is placed in normal (not overstrike), mode.**

### 4.1.6 Select Overlay Planes

MODE : hnum | ØlH, 3AH, hnum

This command allows specific planes to be enabled, thus providing the user with limited overlay capability. (See Fig. 2.Ø5)

The individual bits of hnum cause the individual planes to be enabled or disabled.

hnum

| B | B | B | B |
|---|---|---|---|
| B<br>l<br>i<br>n<br>k | R<br>e<br>d | G<br>r<br>e<br>e<br>n | B<br>l<br>u<br>e |

B = 1 = enabled

B = Ø = disabled

After this command is executed ONLY the enabled planes can be modified by the CRTOS software; this includes erase page, roll, ZOOM, fill, graphics, and characters, foreground and background.

The only exception to this rule is on a full page roll where the hardware is rolling the data. All of the planes are rolled in this case and only the enabled planes are erased.

EXAMPLE: By writing graphics data in red and alphanumerics data in green the graphics data can be erased or scrolled without affecting data on the green plane.

NOTE: Caution should be exercised when using this mode as one could easily get into a situation where commands appear to not be working. To return to normal, issue a:

MODE : F

Since the cursor is normally displayed in white, some residue from the cursor may be left when processing single planes. To prevent this, either turn off the cursor or change the cursor color to the color matching the enabled planes, or enter overstrike mode where the cursor is removed after each character written.

### 4.1.7 Delay

MODE ? number | ØlH, 3FH, hnumber

A delay occurs where number equals the duration of the delay in tenths of seconds. Any delay between 0 and 99.9 seconds can be obtained.

### 4.1.8 Complex Fill

MODE > colnum | ØlH, 3EH, hcolnum

colnum:: = 1 | 2 | 4 | 8

| colnum | color to fill |
|---|---|
| 1 | Blue |
| 2 | Green |
| 4 | Red |
| 8 | Blink |

The cursor should lie "inside" of the polygon to be filled and the border should contain the color selected by colnum. Yellow would contain Red and Green so either could be used for fill. Blue or Blink, however, could not be used. If the area is already filled with that primary color then nothing will happen. If the polygon is not closed or the border color is incorrect, then the entire screen may be filled.

4.1.9 Complex Reverse Fill

MODE < colnum ØlH, 3CH, hcolnum

colnum:: = 1 | 2 | 4 | 8

| colnum | color to fill |
|---|---|
| 1 | Blue |
| 2 | Green |
| 4 | Red |
| 8 | Blink |

This routine is the complement of the Complex Fill Routine and works the same way except that the border must not contain the color represented by colnum and the area to be reverse filled must contain it. This routine will then remove that primary color from that area.

4.1.10 Insert Line

MODE I | ØlH, 49H

The window is scrolled down one line beginning with the line on which the cursor currently resides. An erase line is then done on the line containing the cursor. The effect is to insert a blank line at the current cursor location.

4.1.11 Delete Line

MODE D | ØlH, 44H

The current line is erased and the lines below it are scrolled up one line to replace it. A new blank line will appear at the bottom line of the window.

4.1.12 Insert Character

CTRL W | 17H

The command causes one character space of background color to appear at the current cursor location. The former character under the cursor and the characters to its right are shifted one character to the right. The rightmost character of the line is lost.

4.1.13 Delete Character

CTRL F | Ø6H

This command causes the character under the current cursor location to be deleted. Characters to the right of the cursor are shifted left one character position. One blank character of background color is added on the right hand end of the line.

4.2 Graphic Mode Extension (Option 71)

This option provides for the automatic generation of arcs, circles and rectangles. These figures may also be drawn with all interior dots lit.

4.2.1 Fill On

FILL | MODE F | Ø1H, 46H

This command places the window in fill mode. While in fill mode, arcs, circles and rectangles will be drawn with all interior dots lit.

4.2.2 Fill Off

FILL* | MODE L | Ø1H, 45H

This command places the window in normal (non-fill) mode.

4.2.3 Arc Submode

) coord radius start-deg degrees |

29H, hcoord, hradius, hstart-deg, hdegrees

where

radius ::= number

start-deg ::= number

degrees ::= number

hradius ::= number

hstart-deg ::= number

hdegrees ::= number

# ARC

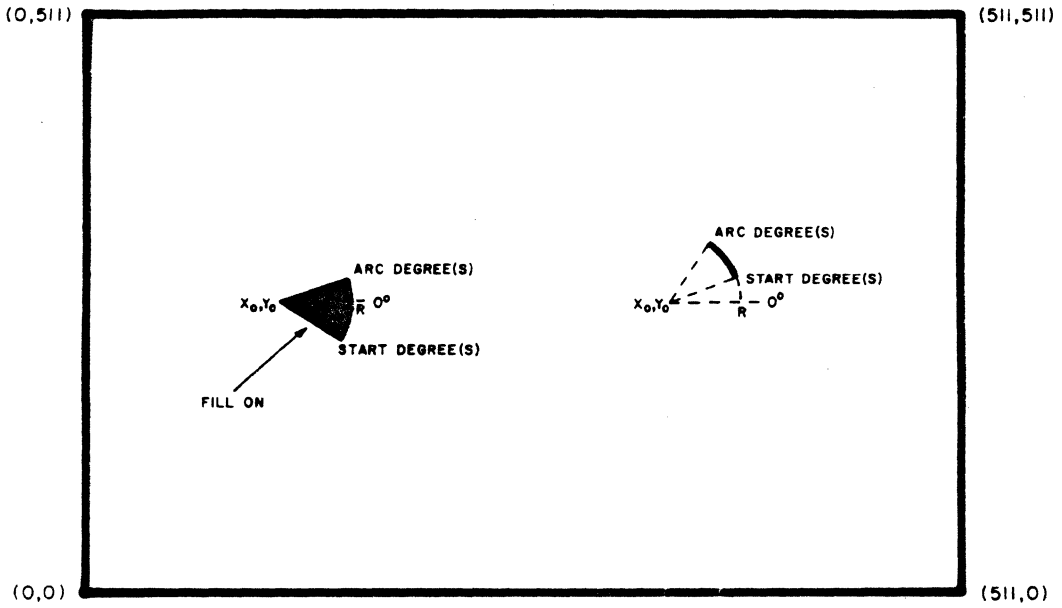



FIGURE 4.1

# CIRCLE

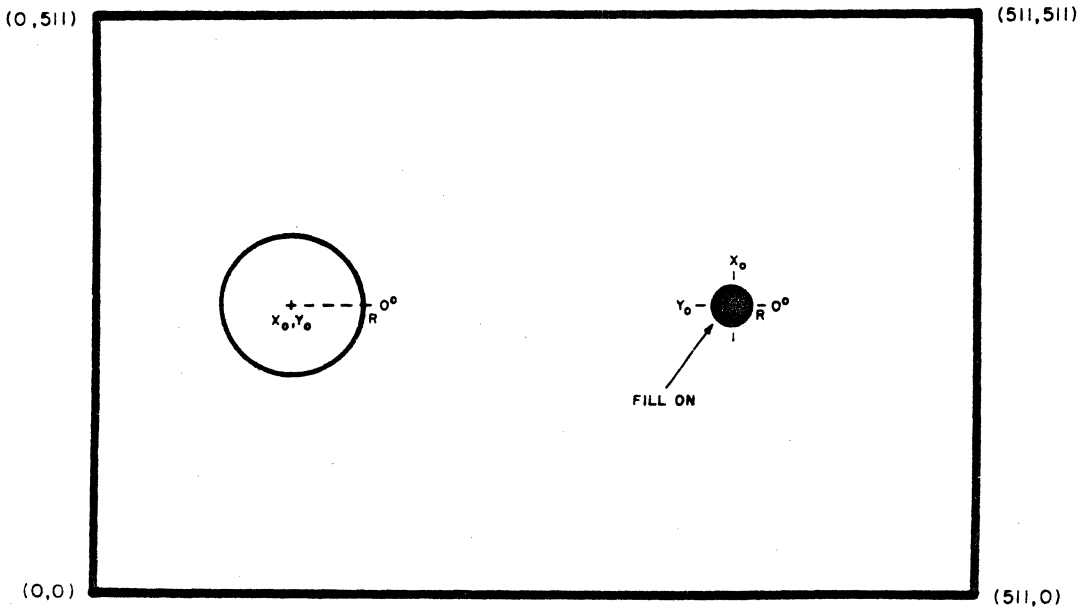



FIGURE 4.2

While in arc submode, the first and each subsequent coord radius start-deg degrees key sequence causes an arc to be drawn with its center at coord, a radius of radius X dot distances, beginning at start-deg for an angle of degrees. After the figure is drawn, the cursor will be at coord. (Note that the normal convention of 0° being horizontally to the right of the center and degrees being measured in the counter-clockwise is observed). If the window is in fill mode, the arc command will draw a pie-shaped wedge. The following key sequence corresponds to Figure 4.1.

```
    ) 345 127 Ø45 Ø2Ø Ø35  FILL 115 127 Ø45 325 Ø55
[ ) 345 255 Ø45 Ø2Ø Ø35  FILL 115 255 Ø45 325 Ø55 ]
```

4.2.4 Circle Submode

CIRCLE coord radius | 2AH, hcoord, hradius

This command places the window in circle plot submode. The first and all subsequent coord radius pairs cause circles to be drawn with centers of coord and with radii of radius X dots. After each pair, the cursor is located at the center of the circle just drawn. If the window is in fill mode, the circle is filled in. The following example corresponds to Figure 4.2.

```
  CIRCLE 115 127 Ø45  FILL 375 125 Ø15
[ CIRCLE 115 255 Ø45  FILL 375 255 Ø15 ]
```

4.2.5 Rectangle Submode

RECT coord coord | 2BH, hcoord, hcoord

This command places the window in rectangle plot submode. The first

# RECTANGLE

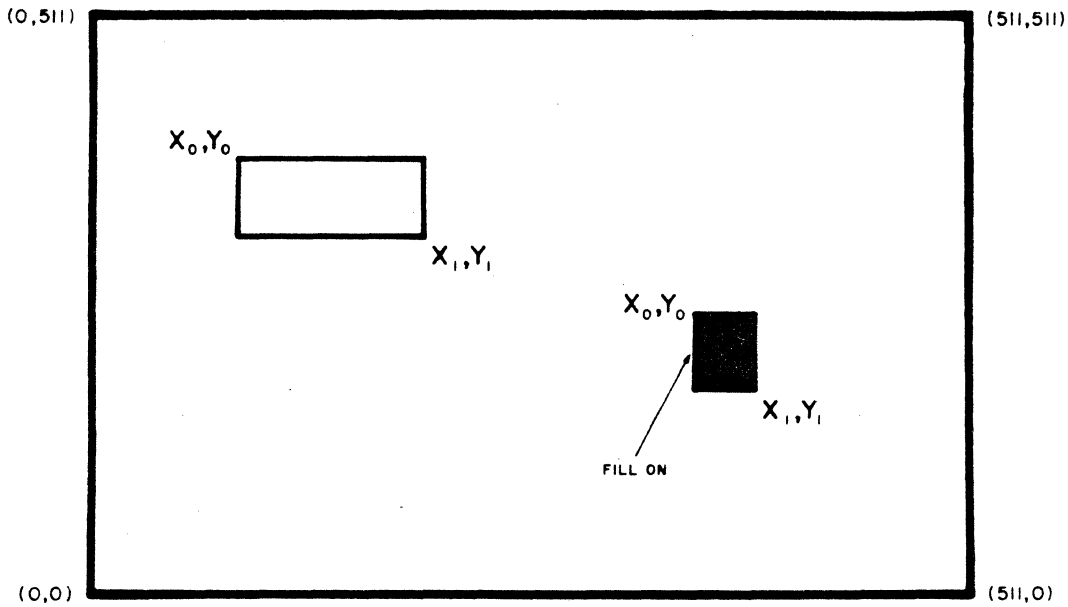



FIGURE 4.3

and each subsequent coord pair causes a rectangle to be drawn with the two coordinates given as opposite corners. The cursor is left at the location of the last coord given. If the window is in fill mode, the interior of the rectangle is filled in. The following example corresponds to Figure 4.3.

```
RECT  Ø85 19Ø  2ØØ 15Ø   FILL  35Ø 127  39Ø Ø85
[ RECT  Ø85 38Ø  2ØØ 3ØØ   FILL  35Ø 255  39Ø 17Ø ]
```

## 4.3 Create and Zoom Processors (Option 73)

This option makes use of the additional memory provided with the option to do special processing. The additional memory is referred to as the "buffer". The buffer is used primarily to store key sequences for future use.

### 4.3.1 Create Buffer

| | | |
|---|---|---|
| CREATE | ESC C | 1BH, 43H |

This command sets up a buffer which stores all subsequent characters as they are entered into the terminal. All mode codes, control characters and plot submodes can be stored in this buffer. However, any ESC code sequence will terminate the function and write an EOF at the end of the buffer. Also, hitting the create key when the light is on will terminate the function.

Multiple subbuffers (up to 255) may be created by separating the data with a CTRL X (EOR character).

4.3.2 View Subbuffer

ESC V hnum | 1BH, 56H, hnum

This command is like the REDRAW command except that only the data between EOR marks hnum and hnum + 1 are sent to AO. hnum is always entered in HEX and can be any number from ØØ to FF.

4.3.3 Kill Subbuffer

ESC K hnum | 1BH, 4BH, hnum

This command allows the data between EOR marks hnum and hnum + 1 to be erased and the following data to be compressed leaving additional room at the end of the buffer.

All subbuffers following the erased subbuffer will be renumbered one number lower than before.

4.3.4 Append to Buffer

ESC Q | 1BH, 51H

This command re-enters the create function. Subsequent characters are added to the buffer beginning at the previous EOF rather than at the first location. The function is terminated as in 4.3.1.

### 4.3.5 Transmit Buffer

XMIT | ESC U | 1BH, 55H

The contents of the buffer up to the EOF are transmitted to logical output device BO by this command.

### 4.3.6 Redraw Buffer

REDRAW | ESC W | 1BH, 57H

The contents of the buffer up to the EOF are transmitted to logical output device AO by this command. (AO is normally assigned to a window).

### 4.3.7 Zoom

ZOOM coord coord | MODE Z coord coord |

ØlH, 5AH, hcoord, hcoord

The zoom command causes the contents of the rectangular area defined by the coordinate pair to be blown up to fill the entire window area, using suitable expansion ratios. Since the original contents of the window are destroyed by this function, it may be useful to use create and redraw to restore the old contents.

### 4.3.8 An Example

The following brief example is intended to illustrate some of the possibilities of using the create and zoom functions.

```
      BOOT CREATE ERASE PAGE RED
PLOT  RECT  . → → → → → → → → → → → → → ↓ ↓ ↓ .  RETURN ↑ ↑ →
PLOT* CYAN    C H R O M A T I C S
      HOME CREATE* ZOOM . → → → → → → → → → → → → → ↓ ↓ ↓ .
      REDRAW
```

# 5. CENTRAL PROCESSING UNIT OPERATING SYSTEM (CPUOS) (Option 61)

The CPU Operating System functions like a system monitor similar to those in most microcomputer systems. It is a local system to aid the operator in such elementary operations as setting and examining memory on a byte-by-byte level. Several utility functions are provided to aid the operator in debugging software by setting program breakpoints and examining CPU registers.

## 5.1 CPUOS Operation

| CPUOS | ESC Z | 1BH, 5AH |
|---|---|---|

The above command causes the CPUOS program to begin execution. The logical device assignments are unnaffected by this command; they remain as they have been set by CRTOS, (see subsection 3.3.6).

CPUOS uses logical device AI, (normally, the keyboard), for commands from the operator and logical device AO, (normally, one of the windows), to display information, including the commands from the operator. Since various colors are used in the displayed information for highlighting, the background color of the window used should be black. Some of the CPUOS commands use logical devices BI and BO for communication. The remaining logical devices are not used.

## 5.2 Command Formats

When CPUOS is ready to accept a new command from the operator, a green sharp symbol (#) is displayed at the left hand edge of the window. The characters coming in AI, which are to make up a command, are displayed

in cyan.

All commands are recognized by their first character, which is followed by zero or more arguments. Most arguments are either machine addresses, (addr), or bytes of information, (byte). These are defined using the conventions of section 1.4 as follows:

hex ::= Ø | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F

addr ::= hex | addr hex

word ::= hex | word hex

which means that addr and byte consist of one or more hexadecimal characters. Addresses and words are automatically padded with leading zeroes, if necessary, to make four characters, while bytes are padded to make two characters. If either a word or byte argument is over its expected length of 4 or 2, respectively, only the low order digits are used. (Note: if multiple arguments of the same type appear in a command, they will be distinguished by subscripts.)

When a command requires more than one argument, the arguments are separated by delimeters, usually a space or comma. A command is usually terminated by a RETURN, but if all possible arguments have been entered, a space or comma sometimes suffices. All delimeters in CPUOS commands are explicitly indicated in section 5.3 by one of the following names:

delim ::= SPACE | ,

fdelim ::= RETURN | SPACE | ,

Note that no delimeter is used between the first character of the command and its first argument.

## 5.3 CPUOS Commands

All CPUOS commands are explained in this section. The possible forms of the command are given using the conventions of sections 1.4 and 5.2.

### 5.3.1 Display Memory

D $\underline{addr_1}$ $\underline{delim}$ $\underline{addr_2}$ $\underline{fdelim}$

The contents of memory from locations $\underline{addr_1}$ through $\underline{addr_2}$ inclusive are displayed on AO. If $\underline{addr_1}$ is omitted, a starting address of zero is used. An example:

D1Ø,25 RETURN

displays hex locations 1Ø through 25, (that's 16 through 31 decimal).

### 5.3.2 End of File

E | E $\underline{delim}$ $\underline{addr}$

This command causes load module format end of file, (see subsection 5.3.7), to be written to logical device BO. If the second format is used, $\underline{addr}$ is stored in the EOF, (for use as a start address on an object program module).

### 5.3.3 Fill Memory

F $\underline{addr_1}$ $\underline{delim}$ $\underline{addr_2}$ $\underline{delim}$ $\underline{byte}$ $\underline{fdelim}$

The contents of memory from locations $\underline{addr_1}$ through $\underline{addr_2}$ inclusive are set to the value of $\underline{byte}$. The memory set must be RAM for this command to have an effect.

### 5.3.4 Execute Program (Go)

G RETURN | G $\underline{addr_1}$ RETURN | G $\underline{addr_1}$ $\underline{delim}$ $\underline{addr_2}$ RETURN |

G $\underline{addr_1}$ $\underline{delim}$ $\underline{addr_2}$ $\underline{delim}$ $\underline{addr_3}$ $\underline{fdelim}$

This command starts program execution at $\underline{addr_1}$. If $\underline{addr_1}$ is omitted, then execution resumes at the previously established program counter.

When present, $\underline{addr}_2$ and $\underline{addr}_3$ cause breakpoints to be set at their respective locations. When a breakpoint is hit in program execution, control returns to CPUOS and the CPU registers at that point are displayed on AO, and the program counter is set to the breakpoint address. Note that setting breakpoints modifies the program, which is restored when the breakpoint is executed. So if no breakpoint is reached, the program will remain modified.

5.3.5 Compute Hex

H $\underline{addr}_1$ $\underline{delim}$ $\underline{addr}_2$ $\underline{fdelim}$

The two hexadecimal numbers equal to $\underline{addr}_1$ and $\underline{addr}_2$ are added and subtracted from one another and displayed over AO. This command is useful in computing the lengths of blocks in memory, relative addresses, etc.

5.3.6 Compare

K $\underline{addr}_1$ $\underline{delim}$ $\underline{addr}_2$ $\underline{delim}$ $\underline{addr}_3$ $\underline{fdelim}$

The block of data stored in locations $\underline{addr}_1$ through $\underline{addr}_2$ is compared, byte for byte, with a block of the same length beginning at location $\underline{addr}_3$. Each byte which gives an unequal comparison, scanning from $\underline{addr}_1$ through $\underline{addr}_2$, causes the following information to be displayed:

> location in the first block, value in the first block, value in the second block

The operator may then type SPACE to continue scanning or RETURN to terminate the comparison. If no unequal comparisons are found, then control returns to CPUOS and the prompt character (#) is displayed.

### 5.3.7 Load

L

This command causes data to be loaded into memory from logical device BI until a load module format end of file is encountered. The data from BI is also in load module format. Each record in load module format consists of a string of ASCII characters in the following form:

:BBAAAAØØDDDDDDDDSS

The colon indicates the start of a string. The B, A, D and S characters must be of the set (Ø,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F). The BB byte is the quantity of data bytes in the string. The AAAA field is the two byte load address where the data is to be stored and the ØØ is a null byte separator. The DDD... (variable length) field contains the data bytes themselves, while the SS byte is the checksum of the entire string except for the colon. A load module format end of file is indicated by a zero length data field, (:ØØAAAAØØSS).

### 5.3.8 Move

M $\underline{addr_1}$ $\underline{delim}$ $\underline{addr_2}$ $\underline{delim}$ $\underline{addr_3}$ $\underline{fdelim}$

The data bytes in locations $\underline{addr_1}$ through $\underline{addr_2}$ are moved, (beginning with the byte at $\underline{addr_1}$), to the locations starting at location $\underline{addr_3}$.

### 5.3.9 Send Nulls

N $\underline{byte}$

The quantity of NULL characters, (ØØH ), given by the value of $\underline{byte}$ are sent to logical output device BO.

5.3.1Ø Dump Memory (Punch)

P $addr_1$ delim $addr_2$ fdelim

The data bytes in memory locations $addr_1$ through $addr_2$ are transmitted to logical output deivce BO in load module format, (see subsection 5.3.7). Note that no end of file is transmitted.

5.3.11 Search (Query)

Q $addr_1$ delim $addr_2$ delim $byte_1$ RETURN |

Q $addr_1$ delim $addr_2$ delim $byte_1$ delim mask fdelim

where

mask ::= $byte_2$

The data bytes from $addr_1$ through $addr_2$ are searched for data equal to $byte_1$. The address of each byte and its contents are listed for one which is equal to $byte_1$. If mask is present, only the bits which are "turned on" in mask, (have a binary value of 1), are compared. The absence of a mask is equivalent to a mask value of FFH, (all ones). The process pauses after each byte. The scan continues if the operator enters SPACE and terminates if RETURN is entered.

5.3.12 Read from Disk

R drv delim trk delim sct delim nms delim addr fdelim

where

drv ::= byte

trk ::= byte

sct ::= byte

nms ::= word

This command performs a direct data read from floppy disk drive drv starting at track trk, sector sct and continuing for nms sectors. The data is loaded into memory starting at address addr.

### 5.3.13 Substitute

S *addr*

Starting at location *addr*, each memory data byte is displayed in yellow followed by a pause. The operator can substitute a new value for the displayed byte by entering a *byte* followed by a SPACE. If no value precedes the space, then that memory location is unaltered. This mode of operation is terminated by a RETURN.

### 5.3.14 Write To Disk

W *drv delim trk delim sct delim nms delim addr fdelim*

This command performs a direct data write to floppy disk drive *drv* starting at track *trk*, sector *sct* and continuing for *nms* sectors. The data is written from memory starting at address addr. (Note: *drv*, *trk* and *sct* are byte quantities and *nms* is a word quantity as defined in 5.3.12).

### 5.3.15 Display Registers

X | X *reg*

where

*reg* ::= A' | F' | B' | C' | D' | E' | H' | L' |
A | F | B | C | D | E | H | L |
S | X | Y | P

If the first form of the command, the CPU registers are simply displayed in the order in which they appear in the definition of *reg*. The second form displays the registers one at a time beginning with the register *reg* and allows them to be modified as in the substitute command, (see 5.3.13). Table 5.1 identifies the registers.

Table 5.1

Eight Bit Registers

| Alternate | Primary |
|---|---|
| A' = ACC | A = ACC |
| F' = Flags | F = Flags |
| B' = Reg B | B = Reg B |
| C' = Reg C | C = Reg C |
| D' = Reg D | D = Reg D |
| E' = Reg E | E = Reg E |
| H' = Reg H | H = Reg H |
| L' = Reg L | L = Reg L |

Sixteen Bit Registers

S = Stack Pointer

X = Reg IX

Y = Reg IY

P = Program Counter

## 5.4 Error Conditions, Escape Codes and Mode Codes

If an error is made during command entry which can be detected by CPUOS, such as a non-hexadecimal character in an address argument, the message "WHAT?" is displayed immediately. The prompt character, (#), is then displayed on the next line, and CPUOS is ready to accept another command.

Escape code processing is done for all input coming from logical device AI. Since this method assures that escape codes are invisible to CPUOS processing, escape codes may occur anywhere, even within CPUOS commands. This allows some functions, (such as reassigning a logical device), to be done directly, without entering CRTOS.

Mode code and control code processing is also allowed while in CPUOS, but these codes are not invisible to CPUOS, so another method is used. Whenever a control character is detected, CPUOS suspends processing of input. CPUOS resumes only when a RETURN character is detected, which terminates the current CPUOS command being processed, if any. This allows window size, colors, etc. to be varied without leaving CPUOS.

## 5.5 Using CPUOS

CPUOS is primarily intended for use with user written programs in assembly language and discussion of this type of use is deferred to the assembler manual. However, it is possible to use CPUOS without access to the assembler to modify the characters displayed. This section will illustrate a way to do this.

### 5.5.1 Memory Organization

The overall structure of the memory address space is shown in Figure 5.1. Note that much of the memory is optional; the basic system can be run with only 6K of PROM and 1K of RAM. (The abbreviation "K" is used to stand for 1Ø24 bytes). The example presented in section 5.5.2 can be done with the basic memory alone, while the example of section 5.5.3 requires either memory Option 22 or 25.

A knowledge of the memory locations used for various purposes is required in order to make effective use of CPUOS. The following list of key address used by CRTOS is sufficient for the examples of 5.5.2 and 5.5.3. For more details see the assembler manual. All address given here are four hexadecimal characters.

| Hex address | Mem. type | Purpose |
|---|---|---|
| 18ØØ | PROM | Beginning of standard character set |
| 1BCØ | PROM | Beginning of standard graphics set |
| 38ØØ | RAM | Window #Ø status table |
| 388Ø | RAM | Window #1 status table |
| 39ØØ | RAM | Window #2 status table |
| 398Ø | RAM | Window #3 status table |
| 3A8Ø | RAM | Input Device assignment table |
| 3A94 | RAM | Output Device assignment table |

### 5.5.2 Changing Fixed Logical Device Assignments

This subsection will show by example how to change the fixed logical device assignments which are initializes by BOOT, (see section 2.2). These assignments are in RAM, along with the variable assignments,

| Decimal Address | Hex Address | Memory Type | Purpose |
|---|---|---|---|
| Ø | ØØØØ | | |
| | | EPROM* | Standard Software |
| 6143 | 17FF | | |
| 6144 | 18ØØ | | |
| | | EPROM* | Standard Alpha and Graphic Char. Sets |
| 8191 | 1FFF | | |
| 8192 | 2ØØØ | | |
| | | EPROM | Optional Software |
| 1Ø239 | 27FF | | |
| 1Ø24Ø | 28ØØ | | |
| | | EPROM | Optional Character Sets |
| 12287 | 2FFF | | |
| 12288 | 3ØØØ | | |
| | | EPROM | DOS Software |
| 14335 | 37FF | | |
| 14336 | 38ØØ | | |
| | | RAM* | Standard Workspace |
| 15359 | 3BFF | | |
| 15360 | 3CØØ | | |
| | | RAM | DOS Workspace |
| 16383 | 3FFF | | |
| 16384 | 4ØØØ | | |
| | | RAM | Special Char. Set (96Ø bytes) |
| 17343 | 43BF | | |
| 17344 | 43CØ | | |
| | | RAM | Buffer |
| 32767 | 7FFF | | |
| 32768 | 8ØØØ | | |
| | | RAM** | Buffer Extension |
| 49151 | BFFF | | |
| 49152 | CØØØ | | |
| | | EPROM** | User Programs or BASIC |
| 65535 | FFFF | | |

* - Memory available in minimal standard system

**- Additional character sets may <u>not</u> be used from these memory areas

Figure 5.1 - Memory Layout

The input and output device assignment tables, beginning at addresses 3AØ and 3A94 respectively, contain lists of addresses which identify the physical devices assigned to the logical devices. Each logical device has two addresses (of two bytes each) associated with it. The first is the variably assigned device and the second is the fixed device. The logical assignments are ordered alphabetically, (AI, BI, CI, DI, EI for the input device assignment table). Thus, the fixed assignment for AI is found at bytes 3A82 and 3A83, while the variable assignment for BO is found at bytes 3A98 and 3A99.

In order to change a fixed assignment, the proper addresses must be loaded into the table. The addresses for the physical input devices begin at address 1E5 , while the physical output devices begin at 205. The addresses are two byte quantities stored in the following order for both lists: Window #Ø, Window #1, Window #2, Window #3, SIO #Ø, SIO #1, PIO #Ø, PIO #1, IEEE-488, Keyboard. Thus, the physical input address for SIO #Ø is stored at bytes 1ED and 1EE, and the output address for Window #1 is stored at bytes 207 and 208.

To change the fixed assignment of BO, (stored at 3A9A), to Window #1, (address stored at 207, the following key sequence could be used:

```
BOOT CPUOS
M 207 SPACE 208 . SPACE 3A9A RETURN
CRTOS
```

After this sequence, CRTOS can be used as a terminal in half or full duplex and all output characters sent to the remote computer will be displayed in Window #1.

### 5.5.3 Modifying the Character Set

This example requires more RAM memory than is supplied with the most basic system. Since the third alternate character set is presumed to be loaded at location 4ØØØ, this is where the new character set will be constructed. The new character set will be identical to the standard, except that the upper and lower case letters will be reversed, effectively changing the meaning of the SHIFT key for alphabetic characters.

Each display character set requires 96Ø bytes, (3CØ hex), of memory, 1Ø for each character. The character formats are stored in order according to their hex codes. To create the modified character set, the entire standard set will first be moved from its permanent location to 4ØØØ. Then the upper case letters will be moved to overlay the lower case, and the lower case letters will be moved to overlay the upper case. When the window is the set to upper character set 3 and the A7 bit is turned on, the upper and lower case letters will be reversed.

```
BOOT CPUOS
M18ØØ,1BBF,4ØØØ RETURN
M194A,1A4E,428A RETURN
M1A8A,1B8E,414A RETURN
CRTOS
MODE S 3
CRTL N
SHIFT  C H R O M A T I C S
```

# 6. ADDITIONAL PROCESSORS

Several additional processors are available as options with the Chromatics series of terminals. These are thouroughly discussed in separate manuals, so only summaries of their operations appear here.

## 6.1 Disk Operating System (DOS) (Option 41)

The Disk Operating System (DOS) allows the user to manipulate data to and from the floppy disk drives on a symbolic name driven, file oriented basis. The DOS commands themselves are symbolic names of disk files. This allows the user to add his own commands to the system at will.

The DOS main program is executed by the following command:

DISK OS | ESC D | 1BH, 44H

The DOS main program accepts its commands from logical device AI and displays them on logical device AO. Escape code processing is always done on AI. The logical assignments of AI and/or AO may be changed with CRTOS before DOS is entered, (see section 3.3.7).

DOS prompts the operator with an asterisk (*) to indicate that a disk command is expected. Disk commands have the following form:

*filename* / *drive* , *arglist* | *filename* , *arglist*

where

*drive* ::= Ø | 1 | 2 | 3 | 4 | 5 | 6

*filename* ::= *name* | *name* . *type*

*type* ::= SYS | SRC | OBJ | DAT | BAS | KIL | ABS

*arglist* ::= *arg* | *arglist* , *arg*

and where *name* is the alphanumeric name of a disk file and *arg* is either a number or an alphanumeric string. The value of *drive* indicates the disk

drive to be searched for filename. If drive = Ø then all drives will be searched in numeric order for filename. If drive is omitted, the last drive used will be searched. The meaning of the type field in the filename is defined below:

SYS
A system command file which is an object file that can only be executed as a DOS command. These files cannot be deleted. They are only listed, with a directory listing, when specifically requested.

SRC
A source file consisting of ASCII codes as generated by the Text Editor. This file type is generated by the Editor for passing source code to the Assembler to produce an object program.

OBJ
A formatted object file consisting of data bytes, load address, checksums and optional execution addresses.

DAT
A data storage file used by the Chromatics BASIC Language Interpreter.

BAS
A program source file used by the Chromatics BASIC Language Interpreter.

KIL
Any file that has been killed, preparing it for deletion from the disk.

ABS
An absolute unformatted binary image file.

To be legitimate, a DOS command must have a filename with type SYS or BAS. (If type is omitted, the drive searches either of these types). SYS files are loaded and executed by the DOS command. BAS files are loaded and the BASIC language processor is executed. For further information, see the DOS operator's manual.

## 6.2 Text Editor (Option 62)

The Text Editor is a developmental tool which allows the user to create and manipulate large source files for use by other programs. The editor is a nonresident routine which may be invoked through the disk operating system, or directly with the following escape function:

TEXT EDIT | ESC X | 1BH, 58H

This command causes the Text Editor to be loaded from disk and executed, logical device assignments are not modified. The Text Editor uses logical device AI for receiving commands and logical device AO for displaying information to the operator. The logical device assignments may be modified using CRTOS, (see section 3.3.7).

The colon (:) is used as a prompt character for commands by the Text Editor. For an explanation of the editor commands, see the Text Editor Manual.

## 6.3 Z-8Ø Disk Assembler (Option 63)

The Z-8Ø Disk Assembler is a developmental tool which allows the user to convert assembly language source text into machine executable object code. The assembler may be invoked through the disk operating system, or directly with the following escape function:

ASMB | ESC A | 1BH, 41H

This function causes the assembler to be loaded from disk and executed with the current logical device assignments, (see section 3.3.7). Communication with the operator is done over logical devices AI and AO. Logical device BO is also used for some output functions. Escape code processing is done on device AI.

The Z-8Ø Disk Assembler is a two pass assembler which accepts free format input. The Z-8Ø symbolic assembly language is described in the Z-8Ø CPU technical manual published by the Zilog or Mostek Corporations. Also see the Chromatics Z-8Ø assembler manual for information on using the assembler.

## 6.4 PROM Programmer (Option 52)

The PROM Programmer enables the user to store his own applications firmware or special graphics character sets in PROM. The PROM Programmer main program is executed by the following escape function:

PROM PGMR | ESC P | 1BH, 5ØH

This function causes the PROM Programmer main program to be loaded from disk and executed with the current logical device assignments. Logical devices AI and AO are used to communicate with the operator. Escape code processing is done on logical device AI.

The PROM Programmer can program all of the bipolar and UV erasable PROMs that are used in the Chromatics CG terminals. These same PROMs are the ones most commonly used in microcomputer systems. The PROM Programmer, together with the other option features available from Chromatics, gives the user the tools needed for a complete and comprehensive program development system. See the PROM Programmer manual for additional details on the use of this system.

## 6.5 BASIC Language Interpreter (Option 64)

Chromatics BASIC is a simple user oriented, high level, symbolic programming language. It consists of English-like statements and simple mathematical expressions combined in a conversational style

for easy formulation of computer problems. The language is quick to learn, easy to use and widely understood. Everything from simple calculations to complex tasks can be efficiently expressed.

The BASIC interpreter resides in PROM and processes programs stored in RAM. The following escape function causes BASIC to be executed:

BASIC | ESC B | 1BH, 42H

This function also intializes the internal memory tables used by BASIC. To re-enter BASIC without re-intializing, the following escape function is available:

ESC E | 1BH, 45H

In either case, the current device assignments are undisturbed. BASIC primarily makes use of logical devices AI and AO for communication with the operator, and escape code processing is done on AI. However, all logical devices are available under BASIC. The BASIC operator's manual gives complete details on the use of BASIC in the Chromatics CG series of terminals.

# A P P E N D I C E S

# A. SUMMARY OF CRTOS FUNCTIONS

This appendix lists all standard and extended functions available from the CRTOS main program. The four table provided facilitate quick referencing.

## A.1 Control Codes

Table A.1 lists all ASCII control codes and their meaning in the Chromatics CG terminal series. Some codes are left unused to allow reassignment of any codes which might cause a conflict in a particular operating environment. The table first gives the ASCII standard name for the code, followed by the key which is modified by CTRL, followed by the meaning assigned to the code. The section and page on which each code is discussed is also given.

## A.2 Escape Codes

Escape code processing is discussed in section 2.3. Table A.2 lists all escape code function key sequence forms together with their meaning and a reference to the appropriate section and page of the manual. In many cases, the key sequence form may provide enough information by itself to make a reference to the text unnecessary.

## A.3 Mode Codes

Mode code functions are used to establish the operating mode of a window. Table A.3 gives all mode code functions along with their meanings and references to the appropriate sections and pages of the

## CONTROL CODES

| Hex | Decimal | ASCII | CTRL | Chromatics Interpretation | Section | Page |
|---|---|---|---|---|---|---|
| ØØ | Ø | NUL | @ | NULL | | |
| Ø1 | 1 | SOH | A | MODE | 3.1Ø.3 | 3-4Ø |
| Ø2 | 2 | STX | B | unused | | |
| Ø3 | 3 | ETX | C | unused | | |
| Ø4 | 4 | EOT | D | unused | | |
| Ø5 | 5 | ENQ | E | SHIFT ↑ (dot up) | 3.6.3.1 | 3-16 |
| Ø6 | 6 | ACK | F | delete character | 1.3.2,4.1.13 | 1-9,4-6 |
| Ø7 | 7 | BEL | G | BELL | 1.3.2 | 1-6 |
| Ø8 | 8 | BS | H | backspace (←) | 3.6.2.5 | 3-14 |
| Ø9 | 9 | HT | I | tabulate | 3.6.2.3 | 3-14 |
| ØA | 1Ø | LF | J | line feed (LF, ↓) | 3.6.2.4 | 3-14 |
| ØB | 11 | VT | K | ↑ (vertical tab) | 3.6.2.6 | 3-14 |
| ØC | 12 | FF | L | erase page, form feed | 3.7.5.1 | 3-22 |
| ØD | 13 | CR | M | RETURN | 3.6.2.2 | 3-13 |
| ØE | 14 | SO | N | A7 on | 3.7.1.1 | 3-17 |
| ØF | 15 | SI | O | A7 off | 3.7.1.2 | 3-18 |
| 1Ø | 16 | DLE | P | unused | | |
| 11 | 17 | DC1 | Q | unused | | |
| 12 | 18 | DC2 | R | unused | | |
| 13 | 19 | DC3 | S | unused | | |
| 14 | 2Ø | DC4 | T | unused | | |
| 15 | 21 | NAK | U | plot off, mode cancel | 3.9.2 | 3-27 |
| 16 | 22 | SYN | V | SHIFT ↓ (dot down) | 3.6.3.2 | 3-16 |
| 17 | 23 | ETB | W | insert character | 1.3.2,4.1.12 | 1-9,4-5 |
| 18 | 24 | CAN | X | end of record for subbuffer | 4.3.1 | 4-1Ø |
| 19 | 25 | EM | Y | SHIFT ← (dot left) | 3.6.3.3 | 3-16 |
| 1A | 26 | SUB | Z | unused | | |
| 1B | 27 | ESC | [ | ESC | 3.3.6,3.1Ø.2 | 3-7,3-38 |
| 1C | 28 | FS | \ | home | 3.6.2.1 | 3-13 |
| 1D | 29 | GS | ] | (cursor right) | 3.6.2.7 | 3-15 |
| 1E | 3Ø | RS | Λ | end file | 1.3.2,4.3.1 | 1-9,4-1Ø |
| 1F | 31 | US | _ | SHIFT (dot right) | 3.6.3.4 | 3-16 |

## ESCAPE CODES

| Escape Code Sequence | Interpretation | Section | Page |
|---|---|---|---|
| ESC A | run assembler | 2.4.6,6.3 | 2-8,6-3 |
| ESC B | run BASIC | 2.4.3,6.5 | 2-7,6-4 |
| ESC C | create buffer | 4.3.1 | 4-8 |
| ESC D | run disk OS | 2.4.4,6.1 | 2-7,6-1 |
| ESC E | warmstart | 2.4.3,6.5 | 2-7,6-5 |
| ESC F | full duplex mode | 3.3.1 | 3-3 |
| ESC G | boot | 2.4.1,3.1.1 | 2-6,3-1 |
| ESC H | half duplex mode | 3.3.1 | 3-3 |
| ESC I logical physical | assign logic. input | 3.3.7 | 3-7 |
| ESC J digit7 | run user function | 2.4.8 | 2-8 |
| ESC K | kill subbuffer | 4.3.3 | 4-11 |
| ESC L | local mode | 3.3.1 | 3-3 |
| ESC O logical physical | asg. logical output | 3.3.7 | 3-7 |
| ESC P | run PROM programmer | 2.4.7,6.4 | 2-8,6-4 |
| ESC Q | append to buffer | 4.3.4 | 4-11 |
| ESC R sio ratecode | set communic. rate | 3.3.2 | 3-3 |
| ESC S sio pscode | set parity & stop b. | 3.3.3 | 3-5 |
| ESC T | run CRTOS | 2.4.1,3.1.2 | 2-6,3-2 |
| ESC U | transmit buffer | 4.3.5 | 4-12 |
| ESC V | view subbuffer | 4.3.2 | 4-11 |
| ESC W | redraw buffer | 4.3.6 | 4-12 |
| ESC X | run editor | 2.4.5,6.2 | 2-8,6-3 |
| ESC Y | trans. cursor pos. | 3.3.4 | 3-6 |
| ESC Z | run CPUOS | 2.4.2,5.1 | 2-2,5-1 |
| ESC ESC | send ESC | 3.3.6 | 3-7 |

manual. The basic form of each mode code is given as an aid to memory.

## A.4 Plot Submodes

The submodes under which the various built-in graphics figures can be generated are listed in Figure A.4. Note that these submodes can only be entered if the window is already in plot mode.

## A.5 XYdelta Codings

The coded movements for the incremental dot mode are summarized in Figure A.5.

## MODE CODES

| Mode code sequence | Interpretation | Section | Page |
|---|---|---|---|
| MODE @ | erase line | 3.7.5.2 | 3-22 |
| MODE A number | set interline space | 3.6.2.8 | 3-15 |
| MODE B | binary coordinates | 3.5.2 | 3-1Ø |
| MODE C digit7 | select color | 3.7.4.1 | 3-21 |
| MODE D | delete line | 4.1.11 | 4-5 |
| MODE E | decimal coordinates | 3.5.1 | 3-9 |
| MODE F | fill on | 4.2.1 | 4-4 |
| MODE G | plot on | 3.9.1 | 3-26 |
| MODE H | write horizontal | 3.7.3.1 | 3-2Ø |
| MODE I | insert line | 4.1.10 | 4-5 |
| MODE J | visible cursor | 3.6.1.1 | 3-12 |
| MODE K | blind cursor | 3.6.1.2 | 3-12 |
| MODE L | fill off | 4.2.2 | 4-6 |
| MODE M | background on | 3.7.4.2 | 3-21 |
| MODE N | background off | 3.7.4.3 | 3-21 |
| MODE O | overstrike | 4.1.3 | 4-2 |
| MODE P | roll off | 4.1.2 | 4-2 |
| MODE Q colnum | set cursor color | 3.6.1.3 | 3-12 |
| MODE R | roll on | 4.1.1 | 4-1 |
| MODE S cset | set upper char. set | 3.7.1.3 | 3-18 |
| MODE T char | test | 3.7.5.3 | 3-22 |
| MODE U coord | move cursor to coord. | 3.6.3.5 | 3-17 |
| MODE V | write vertical | 3.7.3.2 | 3-2Ø |
| MODE W coord coord | set window size | 3.8.1 | 3-23 |
| MODE X number | set char. width | 3.7.2.2 | 3-19 |
| MODE Y number | set char. height | 3.7.2.1 | 3-19 |
| MODE Z coord coord | zoom | 4.3.7 | 4-12 |
| MODE [ | unlatch overstrike | 4.1.5 | 4-3 |
| MODE ] | latch overstrike | 4.1.4 | 4-2 |
| MODE Ø | keyboard sync | 3.8.2.1 | 3-25 |
| MODE 1 | blink on | 3.7.4.4 | 3-21 |
| MODE 2 | blink off | 3.7.4.5 | 3-22 |
| MODE 3 | erase to eol | 3.7.5.4 | 3-23 |

Figure A.3

## MODE CODES

| Mode code sequence | Interpretation | Section | Page |
|---|---|---|---|
| MODE : | select overlay planes | 4.1.6 | 4-3 |
| MODE ? number | delay | 4.1.7 | 4-4 |
| MODE > colnum | complex fill | 4.1.8 | 4-4 |
| MODE < colnum | complex reverse fill | 4.1.9 | 4-5 |

## PLOT SUBMODES

| Command | Submode | Section | Page |
|---|---|---|---|
| ! \| X BAR coord xcoord | X Bar | 3.9.5.1 | 3-29 |
| " \| Y BAR coord ycoord | Y Bar | 3.9.5.2 | 3-30 |
| # coord xcoord | incr. X Bar | 3.9.5.3 | 3-30 |
| $ coord ycoord | incr. Y Bar | 3.9.5.4 | 3-32 |
| % \| DOT coord | Dot | 3.9.5.5 | 3-32 |
| & coord xydelta | incr. Dot | 3.9.5.6 | 3-32 |
| , \| VECTOR coord coord | vector | 3.9.5.7 | 3-36 |
| ( coord coord | concat. vector | 3.9.5.8 | 3-36 |
| ) coord radius start-deg degree | arc | 4.2.3 | 4-6 |
| * \| CIRCLE coord radius | circle | 4.2.4 | 4-8 |
| + \| RECT coord coord | rectangle | 4.2.5 | 4-8 |

Figure A.4

XYdelta Codings

| X movement | Y movement | xydelta | hxydelta |
|---|---|---|---|
| none | none | @ | 4ØH |
| none | 1 up | A | 41H |
| none | 2 up | B | 42H |
| none | 3 up | C | 43H |
| none | 1 down | E | 45H |
| none | 2 down | F | 46H |
| none | 3 down | G | 47H |
| 1 right | none | H | 48H |
| 1 right | 1 up | I | 49H |
| 1 right | 2 up | J | 4AH |
| 1 right | 3 up | K | 4BH |
| 1 right | 1 down | M | 4DH |
| 1 right | 2 down | N | 4EH |
| 1 right | 3 down | O | 4FH |
| 2 right | none | P | 5ØH |
| 2 right | 1 up | Q | 51H |
| 2 right | 2 up | R | 52H |
| 2 right | 3 up | S | 53H |
| 2 right | 1 down | U | 55H |
| 2 right | 2 down | V | 56H |
| 2 right | 3 down | W | 57H |
| 3 right | none | X | 58H |
| 3 right | 1 up | Y | 59H |
| 3 right | 2 up | Z | 5AH |
| 3 right | 3 up | [ | 5BH |
| 3 right | 1 down | ] | 5DH |
| 3 right | 2 down | ^ | 5EH |
| 3 right | 3 down | _ | 5FH |
| 1 left | none | h | 68H |
| 1 left | 1 up | i | 69H |
| 1 left | 2 up | j | 6AH |
| 1 left | 3 up | k | 6BH |
| 1 left | 1 down | m | 6DH |
| 1 left | 2 down | n | 6EH |
| 1 left | 3 down | o | 6FH |
| 2 left | none | p | 7ØH |
| 2 left | 1 up | q | 71H |
| 2 left | 2 up | r | 72H |
| 2 left | 3 up | s | 73H |
| 2 left | 1 down | u | 75H |
| 2 left | 2 down | v | 76H |
| 2 left | 3 down | w | 77H |
| 3 left | none | x | 78H |
| 3 left | 1 up | y | 79H |
| 3 left | 2 up | z | 7AH |
| 3 left | 3 up | { | 7BH |
| 3 left | 1 down | } | 7DH |
| 3 left | 2 down | ~ | 7EH |
| 3 left | 3 down | DEL | 7FH |

Figure A.5

## B. ELECTRICAL AND MECHANICAL SPECIFICATIONS - STANDARD SYSTEM

### B.1 General

Power: 105-125 volts, 60 Hz, 600 watts.

Temperature: +10°C to +45°C operating, -30°C to +70°C storage.

Humidity: 0-95% noncondensing.

Package Color: Light beige (Federal Standard 26521) and brown (Federal Standard 20140).

X Radiation: Less than 0.5 milliroentgen per hour at a distance 2 inches from all exterior surfaces.

### B.2 Video Display

Screen Size and Format:

| Model | Diagonal Measure | Phosphor Area Inches | Phosphor Area Sq. Inches | Usable Display Area Inches | Usable Display Area Sq. Inches | Format: Displayable and Addressable Dots |
|---|---|---|---|---|---|---|
| 1398 | 13" | 8.15×10.87 | 88.75 | 7×10 | 70 | 512×256 |
| 1399 | 13" | 8.15×10.87 | 88.75 | 7×10 | 70 | 512×512 |
| 1598 | 15" | 9.37×12.44 | 117.14 | 8.25×11.25 | 94.87 | 512×256 |
| 1599 | 15" | 9.37×12.44 | 117.14 | 8.25×11.25 | 94.87 | 512×512 |
| 1998 | 19" | 11.7×15.61 | 182.67 | 10.25×14.5 | 148.63 | 512×256 |
| 1999 | 19" | 11.7×15.61 | 182.67 | 10.25×14.5 | 148.63 | 512×512 |

Refresh Rate: 60 times/second noninterlaced, synchronized to 60 Hz line frequency.

Color Levels: 8 foreground and 8 background - red, green, blue, magenta, cyan, yellow, white and black.

Convergence: 9 sector, with each sector individually converged from front drawer accessible controls.

Controls: Brightness, focus, convergence, on/off.

Deflection: Magnetic.

Focus: High voltage electrostatic.

Blink Rate: 1.9 Hz, cursor and dots, foreground and/or background.

Cursor: 4 each, 1 per window; programmable in position, color and visibility (on or off). Expands vertically to match character height when character Y magnification is not equal to 1.

## B.3 Processor Memory

CPU: Z-80.

Refresh Memory: Models 1398, 1598 and 1998 - 65,536 bytes of dynamic RAM. Models 1399, 1599 and 1999 - 131,072 bytes of dynamic RAM.

Program Memory: 8192 bytes of EPROM and 1024 bytes of RAM for base routines and CRT Operating System.

## B.4 Peripherals

Keyboard: Detachable; capacitive; 128 stepped keys; visual light feedback on alternate action mode keys; 2 key rollover; automatic repeat; cursor pad; numeric pad; special function keys; sculptured keys in main keyboard section.

## B.5 Software

Processors: CRT Operating System with full, local and half duplex modes as well as ESCAPE code processing and device assignment capability.

## B.6 Display Functions

| | |
|---|---|
| Graphic Mode: | DOT, Incremental DOT, X Bar, Incremental X Bar, Y Bar, Incremental Y Bar, Vector, Concatenated Vector. |
| Alphanumeric Mode: | ASCII and special characters available from RAM or EPROM positioned to any dot position on screen. |
| Coordinate Entry: | Decimal digits, binary codes or cursor position. |
| Character Format: | 96 ASCII upper and lower case 5×7 dot matrix characters and 96 6×10 dot matrix characters in EPROM memory. |
| Character Magnification: | Individually settable in X and/or Y to any integer multiplier. |
| Cursor: | Programmable in color, position and visibility (on or off). |
| Character Interline Spacing: | Variable, up to 255 raster lines. |
| Character Write Direction: | Vertical or Horizontal. |
| Character Resolution: | 512×256 dot resolution - 85 characters/line by 25 lines/page. 512×512 dot resolution - 85 characters/line by 51 lines/page. |
| Control Functions: | Cursor Up, Down, Left and Right (character and dot spacing), Erase Page, Erase Line, Home, Tab, Carriage Return, Line Feed and Backspace. |
| Windows: | 4 each, individually programmable in size, position and all the above functions including a separate cursor for each window. Each window is a physical output device |

and may receive data from any physical input device or logical output device when properly assigned.

## B.7 Interfaces

Serial: Serial I/O port - asynchronous; independently programmable from 110 to 31250 baud, (9600 baud highest standard rate); single stop bit, (programmable to 1.5 or 2); TTL and RS-232C interface with busy output lines (clear to send status line also included in RS-232C interface).

## C. ELECTRICAL AND MECHANICAL SPECIFICATIONS - OPTIONS

### C.1 General

Power: Option 11 - 205-250 volts.

Option 12 - 50 Hz.

### C.2 Video Display

Refresh Rate: With Option 12 installed - 50 times/second, noninterlaced, synchronized to 50 Hz line frequency.

Character Format: Option 21 - 192 additional customer defined special graphic characters programmed in EPROM memory.

Option 27 - 7×9 dot matrix ASCII character and 8×10 dot matrix graphic character formats.

With optional additional RAM memory, additional user defined character sets may be loaded into memory, (960 bytes of RAM required per set).

### C.3 Memory

Program Memory: Option 22 - Memory card with 16,384 bytes of RAM; additional space for 16,384 bytes of RAM and 16,384 bytes of UV erasable PROM.

Option 23 - 16,384 bytes of dynamic RAM, (IC's only).

Option 24 - 2048 bytes of UV PROM, (IC's only).

Option 25 - Memory card with 8192 bytes of RAM; additional space for 8192 bytes of RAM and 16,384 bytes of UV erasable PROM.

Option 26 - 8192 bytes of dynamic RAM. (IC's only).

Option 28 - PROM Expander card with space for 6144 bytes of PROM. (Note: this option is never required if Option 33 or 41 is included in the system).

Floppy Disk: Option 41 - Floppy disk controller with DOS software. Can support up to 6 standard drives or 2 Minifloppy drives.

Option 42 - Standard floppy disk drive with 250,000 bytes of storage and 250,000 bits/second transfer rate.

Option 43 - Dual floppy disk drive with twice the capacity of Option 42.

Option 44 - Minifloppy disk drive with 80,000 bytes of storage and 125,000 bits/second transfer rate.

Option 45 - Dual Minifloppy disk drive with twice the capacity of Option 44.

PROM Programmer: Option 52 - PROM Programmer, interface and software for programming bipolar and UV erasable PROMs.

## C.4 Software

Processors: Option 61 - CPU Operating System that includes Display Memory; Fill; Go (with breakpoints); Hex Arithmetic; Load Hex; Move; Write Hex; Substitute; Direct Disk Read; Direct Disk Write; Search; Compare and Display CPU Registers.

Option 62 - Disk Text Editor with 11 editting commands and functions including disk file operations.

Option 63 - Z-80 Disk Assembler that includes 9 operator commands; pseudo-ops; symbolic addressing and file concatenation capabilities.

Option 64 - BASIC Language Interpreter with 24 keyword program statements; 17 editting and command statements; 19 mathematical functions; 6 string functions; 17 file operations and 14 arithmetic operations.

Option 73 - ZOOM and Buffer Processor - ZOOM expands any rectangular area to the entire window area, (automatically scaled to closest fit). Buffer Processor has CREATE and REDRAW functions. Codes are input into the buffer while in CREATE mode. The REDRAW function reexecutes the buffer to recreate the original display. The buffer may be transmitted using the XMIT function or saved on disk by the Disk Operating System.

## C.5 Display Functions

Graphic Mode: Option 71 - Extended graphic functions (arc generation, circle generation, rectangle generation and solid object auto fill).

Alphanumeric Mode: Option 72 - Extended alphanumeric functions, (roll lines within window, insert line, delete line, insert character, delete character, overstrike).

## C.6 Interfaces

Serial:

Option 31 - Second serial I/O port - asynchronous, independentally programmable from 110 to 31,250 baud, (9600 baud highest standard rate), single stop bit (programmable to 1.5 or 2), TTL, RS-232C, RS-422 and 20 mA current loop interfaces with busy output lines (additional clear to send status line included in RS-232C interface).

Option 32 - RS-422 and 20 mA current loop inerface for standard SIO port #1.

Parallel:

Option 33 - Parallel I/O port, 2 each - 8 bits (programmable to a single 16 bit I/O port), TTL compatible with 2 handshakes on each 8 bit port, 416,000 bytes/second transfer rate maximum with DMA option (208,000 words/second transfer rate with 16 bit operation and DMA option).

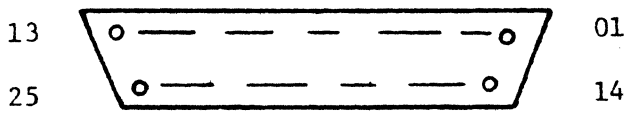
Option 34 - DMA controller - provides high speed transfers from memory to memory, I/O to memory, memory to I/O and I/O to I/O. Maximum transfer rate is 416,000 bytes per second.

Option 35 - IEEE-488 (GPIB) interface and controlling software.

## D. EXTERNAL INTERFACE SPECIFICATIONS

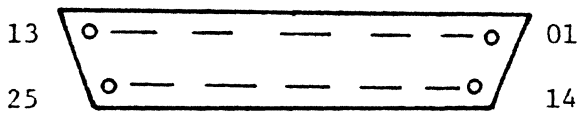
This appendix gives specifications for the RS 232 and RS 422 external interface couplers. The numbering for the pins is given in Figure D.1, (both types use the same numbering).

Table D-1 gives the meanings of the pins for both RS 232 connectors J3 and J4.




Connector Rear Chassis View J3 & J4

Figure D-1




Connector Rear Chassis View J7

Figure D-2

## Table D-1

## RS 232 Connectors

| | | | |
|---|---|---|---|
| 1 | [redacted] | 14 | |
| 2 | [redacted] | 15 | |
| 3 | [redacted] | 16 | |
| 4 | [redacted] | 17 | |
| 5 | [redacted] | 18 | |
| 6 | | 19 | |
| 7 | [redacted] | 2Ø | [redacted] $\overline{\text{Busy}}$ (Output) |
| 8 | | 21 | |
| 9 | | 22 | |
| 1Ø | TTL $\overline{\text{Send Data}}$ (Output) | 23 | |
| 11 | TTL $\overline{\text{Receive Data}}$ (Input) | 24 | |
| 12 | | 25 | TTL $\overline{\text{Busy}}$ (Output) |
| 13 | | | |

## Table D-2

### RS 422 & 20 ma. Current Loop Connector

| | | | |
|---|---|---|---|
| 1 | RS 422-1 $\overline{\text{Busy}}$ (Output) | 14 | RS 422-1 Busy (Output) |
| 2 | RS 422-Ø $\overline{\text{Busy}}$ (Output) | 15 | RS 422-Ø Busy (Output) |
| 3 | RS 422-1 $\overline{\text{Send Data}}$ (Output) | 16 | RS 422-1 Send Data (Output) |
| 4 | RS 422-Ø $\overline{\text{Send Data}}$ (Output) | 17 | RS 422-Ø Send Data (Output) |
| 5 | RS 422-1 $\overline{\text{Receive Data}}$ ( Input) | 18 | RS 422-1 Receive Data (Input) |
| 6 | RS 422-Ø $\overline{\text{Receive Data}}$ ( Input) | 19 | RS 422-Ø Receive Data (Input) |
| 7 | Port 1 Select Bit A | 2Ø | Port 1 Select Bit B |
| 8 | Port Ø Select Bit A | 21 | Port Ø Select Bit B |
| 9 | Select Ground | 22 | CL-Ø Receive Data "+" |
| 1Ø | CL-Ø Receive Data "-" | 23 | CL-1 Receive Data "+" |
| 11 | CL-1 Receive Data "-" | 24 | CL-1 Send Data "-" |
| 12 | CL-1 Send Data "+" | 25 | CL-Ø Send Data "-" |
| 13 | CL-Ø Send Data "+" | | |

SELECT LOGIC

| | B | A |
|---|---|---|
| CL | Pin 9 | Pin 9 |
| RS422 | Pin 9 | nc |
| TTL | nc | Pin 9 |
| RS232 | nc | nc |

nc = no connection

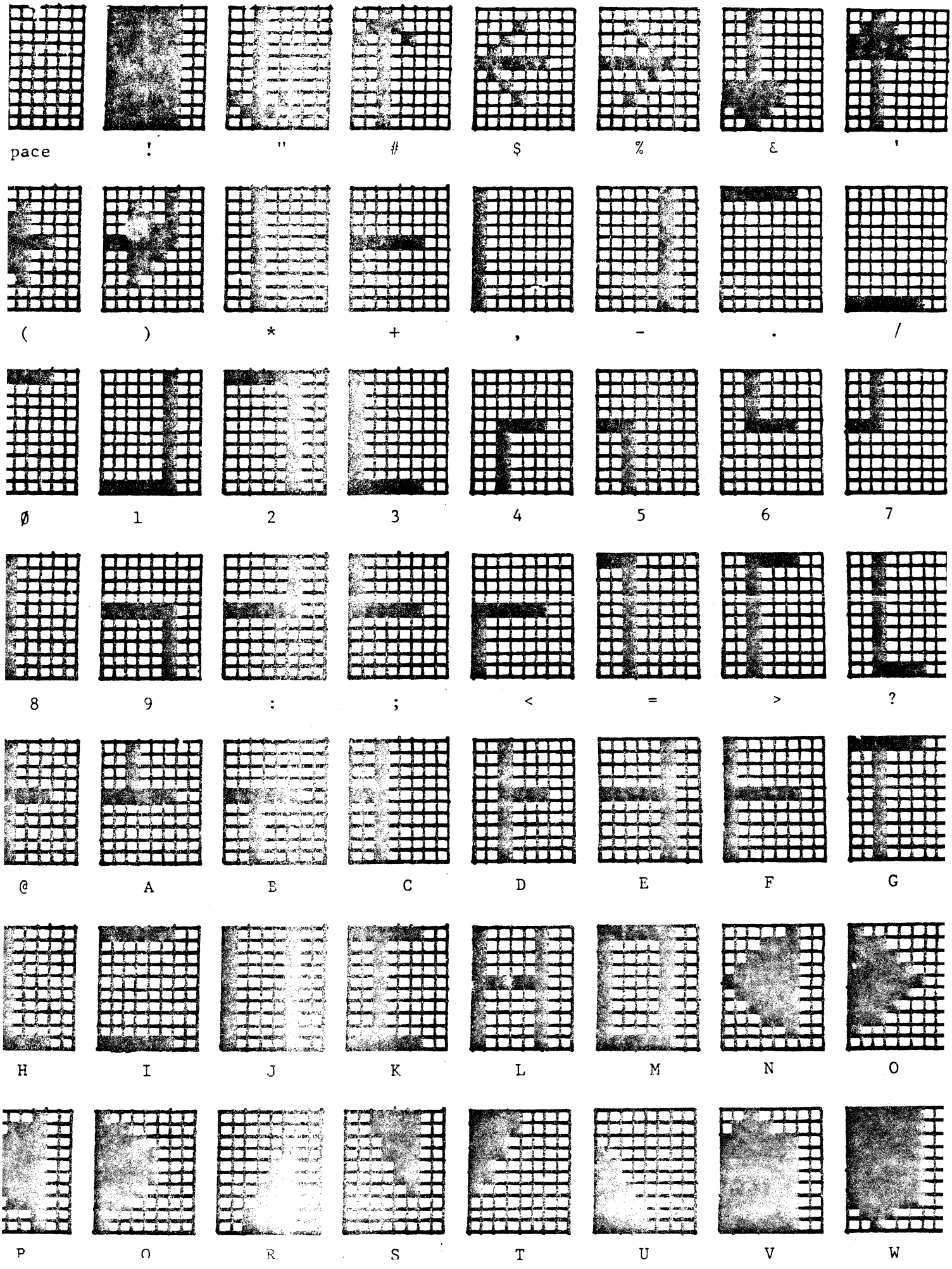
Connectors J4 and J3 may be selected as shown in Figure D-3 to be compatible with either RS-232 or TTL specifications. That is, to use J4 (SIO #Ø) as a TTL port, connect pin 8 to pin 9 on connector J7. To use J3 (SIO #1) as a TTL port, connect pin 7 to pin 9 on J7.

RS-422 and current loop connections are both done on connector J7 for SIO #Ø and SIO #1 as shown in Table D-2. Note that if RS-422 or current loop is selected for SIO #Ø or SIO #1, the corresponding RS-232 connector (J4 or J3) is not used. Table D-2 and Figure D-3 indicate that RS-422 operation can be selected for SIO #Ø and/or SIO #1 by connecting pin 21 and/or pin 20 (respectively) to pin 9 or connector J7. Current loop operation may be selected by connecting pins 8 and 21 to 9 for SIO #Ø and 7 and 20 to 9 for SIO #1.

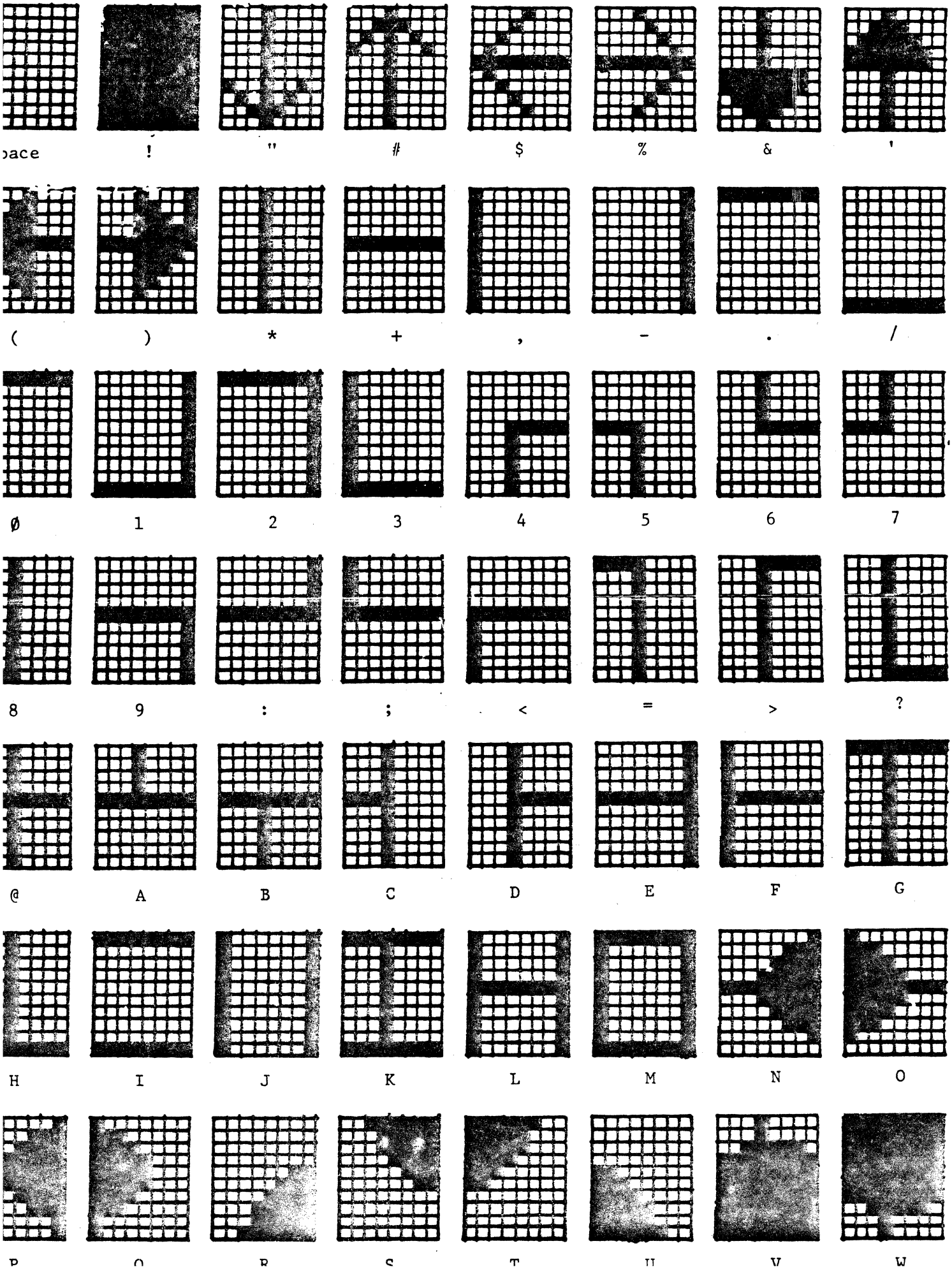
# APPENDIX E

## Special Graphic Character Fonts

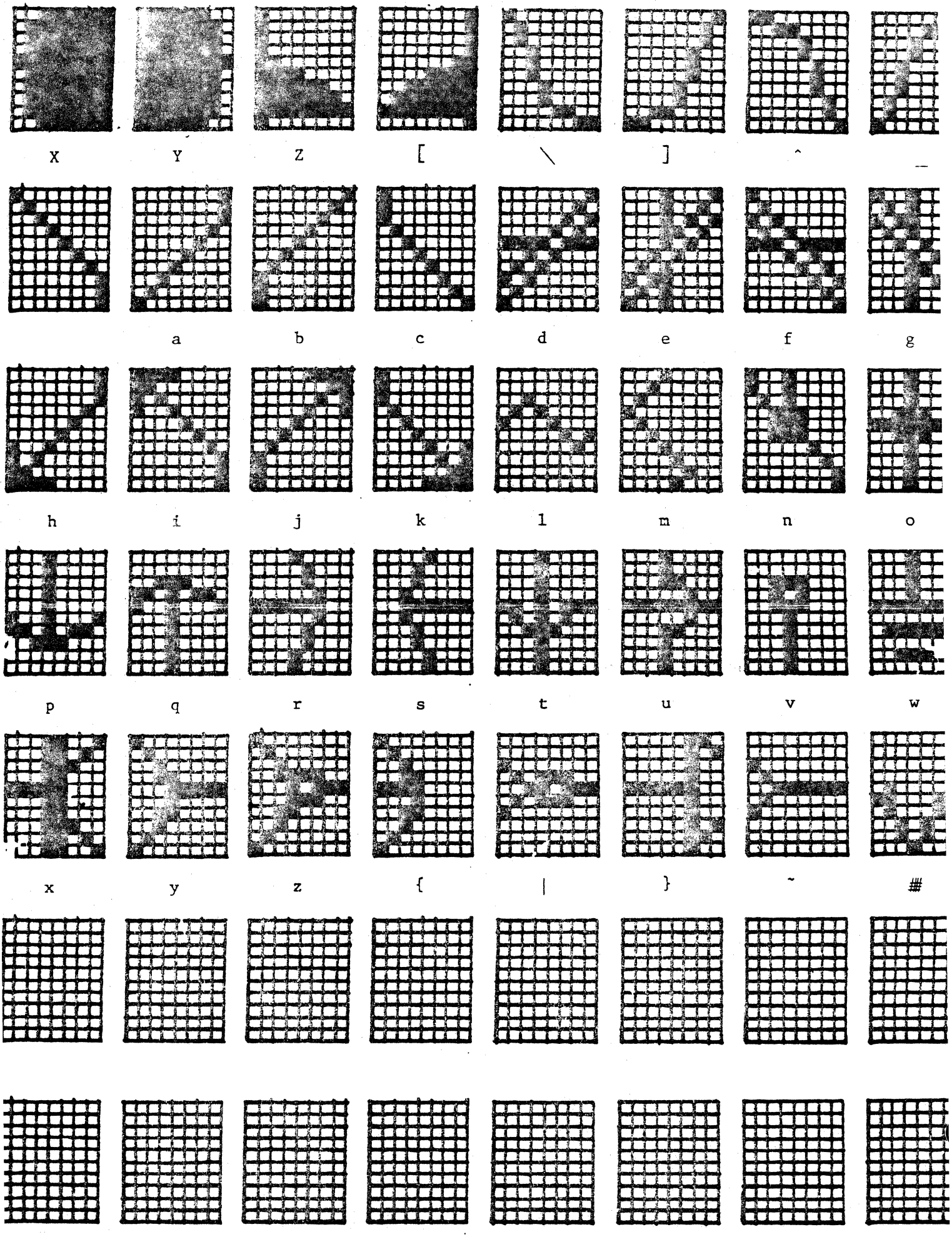
# 5 X 7 Special Graphic Character Set



/ X 9 Special Graphic Character Set



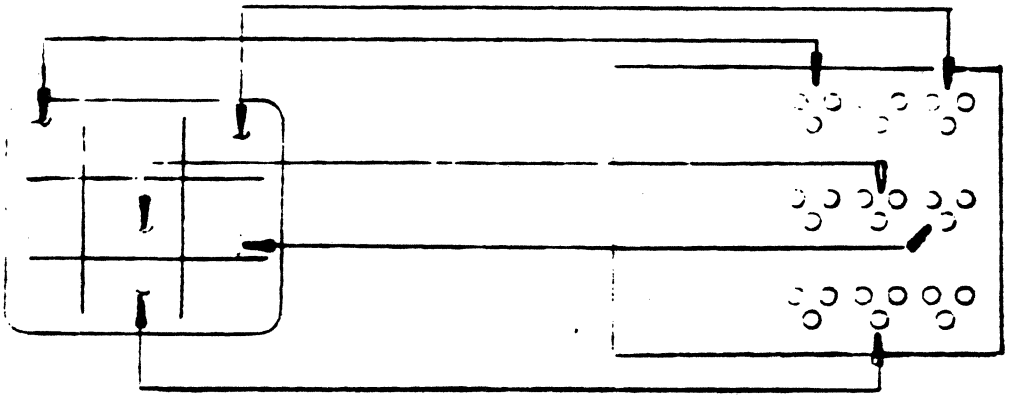
7 X 9 Special Graphic Character Set



## F. HOW TO ADJUST CONVERGENCE

Chromatics' proprietary method of convergence control makes touch-up adjustment of convergence easy. Now anyone can keep their Chromatics display looking as good as new!

The front drawer accessible convergence controls are arranged into nine groups of three controls per group; one control for blue, one for red and one for green. The nine groups correspond to nine areas or sectors of the display screen, matching exactly as you look at them (see diagram below).



Display Screen Front Drawer Controls

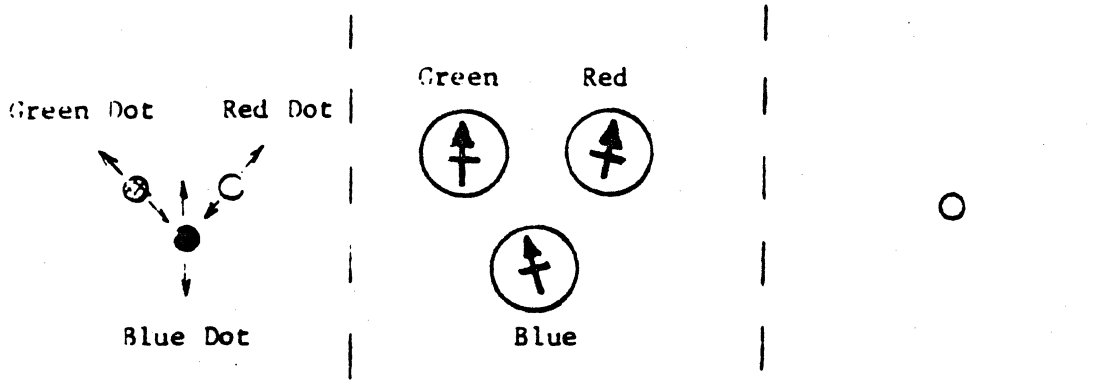
ADJUSTING CONVERGENCE: CENTER FIRST; CORNERS LAST

The Chromatics display computer has a built in test function which can be used to aid in adjusting convergence. Simply depress the "Test" key and then the "." key. The screen will then fill with dots. Dots are a good test pattern to adjust with because they show every minor error in convergence.

Now, select a dot in the <u>center</u> of the display screen <u>first</u> and adjust the three controls in the center group of convergence controls so that all three

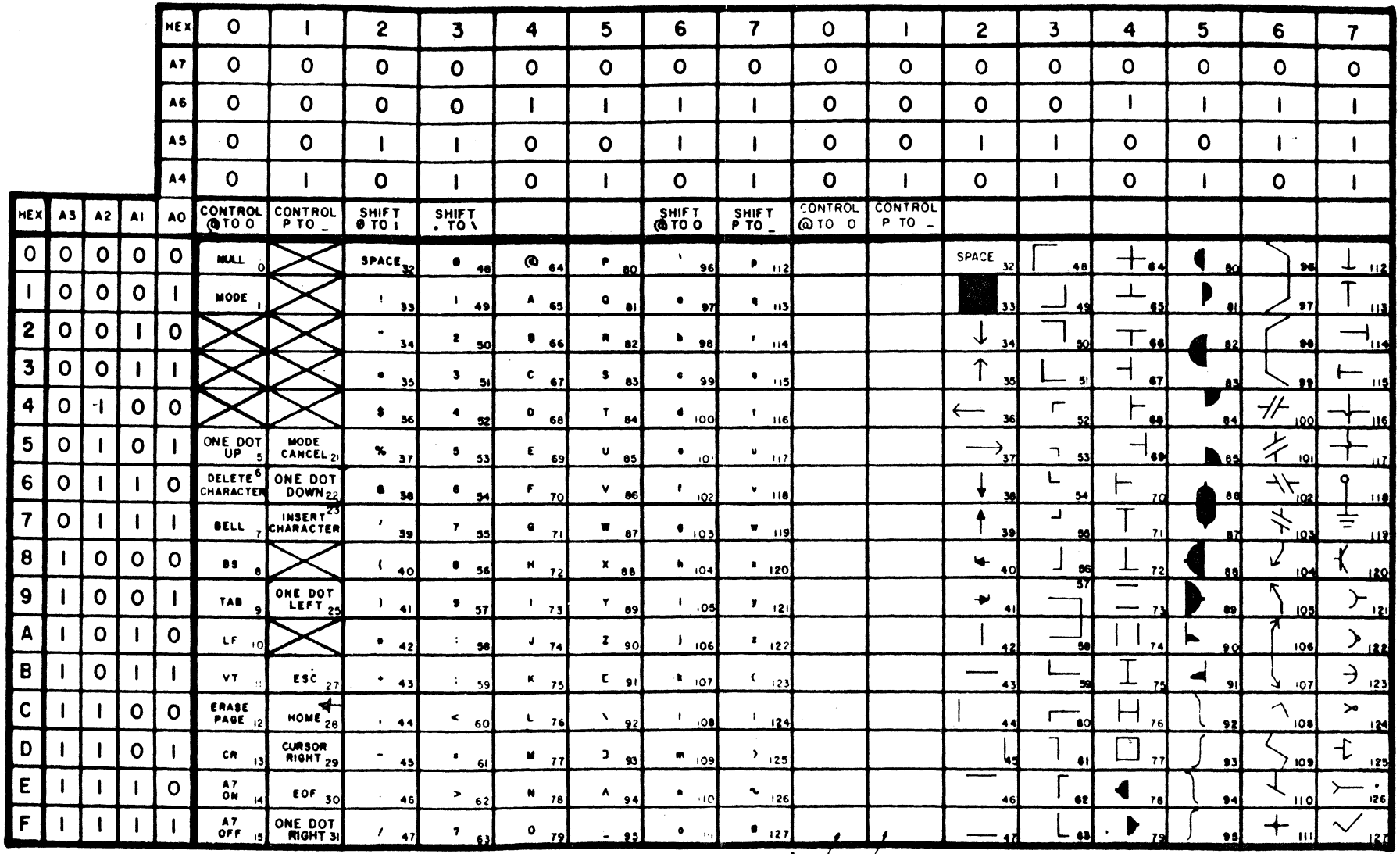
color dots align or "converge" on the same point (which then makes a white dot).

Proceed to adjust the top center, bottom center, left center and right center sectors (in any order). Always do the corners last.




MISCONVERGED DOTS CONTROLS CONVERGED DOT

# ASCII CODE ASSIGNMENT




NOTE: ENTRIES WITH ✕ INDICATE UNUSED ANSI ASCII CODES.

FIGURE 1.3

# I. INDEX

This index lists many of the key terms and phrases used in the manual in alphabetical order. References are given by section and page. Also see Appendix A.

extension cable on the data communication equipment is permitted. An extension cable with a male connector shall be provided with the data terminal equipment. The use of short cables (each less than approximately 50 feet or 15 meters) is recommended; however, longer cables are permissible, provided that the resulting load capacitance ($C_L$ of Fig. 2.1), measured at the interface point and including the signal terminator, does not exceed 2500 picofarads. (See section 2.4 and 6.5.)

3.1.1 When additional functions are provided in a separate unit inserted between the data terminal equipment and the data communication equipment (See section 1.7), the female connector, as indicated above shall be associated with the side of this unit which interfaces with the data terminal equipment while the extension cable with the male connector shall be provided on the side which interfaces with the data communication equipment.

| Pin Number | Circuit | Description |
|---|---|---|
| 1 | AA | Protective Ground |
| 2 | BA | Transmitted Data |
| 3 | BB | Received Data |
| 4 | CA | Request to Send |
| 5 | CB | Clear to Send |
| 6 | CC | Data Set Ready |
| 7 | AB | Signal Ground (Common Return) |
| 8 | CF | Received Line Signal Detector |
| 9 | – | (Reserved for Data Set Testing) |
| 10 | – | (Reserved for Data Set Testing) |
| 11 | | Unassigned (See section 3.2.3) |
| 12 | SCF | Sec. Rec'd. Line Sig. Detector |
| 13 | SCB | Sec. Clear to Send |
| 14 | SBA | Secondary Transmitted Data |
| 15 | DB | Transmission Signal Element Timing (DCE Source) |
| 16 | SBB | Secondary Received Data |
| 17 | DD | Receiver Signal Element Timing (DCE Source) |
| 18 | | Unassigned |
| 19 | SCA | Secondary Request to Send |
| 20 | CD | Data Terminal Ready |
| 21 | CG | Signal Quality Detector |
| 22 | CE | Ring Indicator |
| 23 | CH/CI | Data Signal Rate Selector (DTE/DCE Source) |
| 24 | DA | Transmit Signal Element Timing (DTE Source) |
| 25 | | Unassigned |

**Figure 3.1**

**Interface Connector Pin Assignments**